UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Sistema de Respuesta Automática Basado en Recursos Semánticos

PROYECTO FIN DE CARRERA
INGENIERÍA TÉNICA DE INFORMÁTICA DE GESTIÓN

Autor: David Celaá Morales
Tutor: Julio Villena Román

Abril 2010

Título: Sistema de Respuesta automática basado en SPARQL

Autor: David Celaá Morales

Tutor: Julio Villena Román

EL TRIBUNAL

Presidente: Jesús Arias Fisteus

Secretario: Antonio de la Oliva

Vocal: Inmaculada Tomeo-Reyes

Realizado el acto de defensa del Proyecto Fin de Carrera el día 22 de Abril de 2010 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

Fdo: Presidente Fdo: Vocal Fdo: Secretario

Breve descripción del trabajo:

El objeto del proyecto consiste en la realización de un sistema de respuesta automática que sea capaz de “entender” preguntas en lenguaje natural y devolver una respuesta correcta al usuario.

Las respuestas del sistema se basan en una base de datos de información semántica, pudiendo por tanto dar contestación exclusivamente a preguntas cuya información esté contenida en la base de datos.

El objetivo real, y necesario para la creación del sistema, es diseñar un algoritmo versátil y eficiente que permita traducir una consulta en lenguaje natural a una expresión de consulta en un lenguaje de base de datos que permita acceder a la información para obtener los datos requeridos.

Para esta conversión se requiere de unos diccionarios de traducción específicos dependientes de los datos que se alberguen en la base de datos.

Además del diseño del sistema, se realizará una evaluación de aciertos en las respuestas ofrecidas por el sistema en forma de porcentaje, también una breve explicación descriptiva sobre cada método empleado, y como se ha creado la batería de preguntas que ha generado cada calificación.

Agradecimientos:

En primer lugar agradecer a aquellos sin los cuales obviamente no hubiera podido llegar a este punto, no solo llegar sino comenzar esta etapa que culmina con la redacción de este escrito, tan completa y con tan grandes momentos como ha sido la universitaria. Aunque ellos saben quién son creo que he de mencionarles para que quede claro que sin mis padres no hubiera podido llegar tan lejos…

Gracias también a mi tutor por hacerme un hueco en su estresante agenda rebosante de citas, y perder su poco tiempo libre conmigo, y esta redacción aguantando mis consultas y dudas.

Como pauta final que es éste escrito quiero también recordar esos grandes instantes que transcurrían a altas horas en la biblioteca haciéndola un lugar agradable, aún en esos tiempos tan oscuros como época de exámenes y a sabiendas de la que se nos cernía.

Sobra recalcar las "noches de kebap" que te recargaban lo suficiente para aguantar el tiempo que hiciera falta en buena compañía; y posteriormente nuevos acompañantes vendrían en las sucedáneas "comidas de kebap" donde aunque no comieran kebap y miraran con recelo acompañaban esos ratos de relax. Otros merecedores de mención son los grandes sujetos que aquí y allá pululaban con sus locuras a cuestas o conversaciones. A todos, Gracias.

Además, por supuesto no he de olvidar extracadémicamente a los que conseguían extraerme de entre línea y línea de código para llevarme a dar un paseo (bien fuera por aire, por tierra entre “adictos”, entre dianas y anís o entre zombis y cuartos de baños), creando así nuevas aventuras que recordar y que te daban ánimos para seguir en los tan continuados momentos de desánimo mientras conseguía encarrilar este trabajo a su fin.
También agradecer a los que cual estrella polar me orientaban y me enfilaban en la dirección correcta cuando navegaba sin rumbo ni objetivo por los entramados de la programación, y todo ello sin levantarme de la silla.

Por penúltimo aunque ya hayan sido mencionados o no con anterioridad, a los que de algún modo u otro han colaborado, ayudado o me han tirado de las orejas para que no abandonara y terminara, valiéndose de sus mejores caras, gestos o palabras a los que no se les puede decir no.

Y por último, agradecer nuevamente a mi tutor todo el tiempo que ha desperdiciado en mí, y en leer y releerse hasta el punto que podría hacer la presentación él mejor que yo. Y pedirle perdón por los momentos de desánimo que le he ocasionado cuando las cosas no salían como preveía ;)
Gracias ☺

Índice del documento

Breve descripción del trabajo:....2
Agradecimientos:....3
Índice del documento....I
Índice de ilustraciones....III
1. Introducción....1
1.1 Marco del proyecto....1
1.2 Objetivos....2
1.3 Contenido de la memoria....3
2. Estado del arte....4
2.1 Ingeniería lingüística....4
2.1.1 Definición....4
2.1.2 El lenguaje y las necesidades de la ingeniería lingüística....5
2.1.3 Procesamiento del lenguaje natural....6
2.1.4 Problemas en el procesamiento del lenguaje natural....14
2.2 Sistema de respuesta automática a preguntas....15
2.2.1 ¿Qué es la búsqueda de respuestas?....15
2.2.2 Arquitectura de un sistema de respuesta automática....16
2.2.3 Características de los sistemas QA....18
2.2.4 Desarrollo mundial....18
2.3 Procesamiento del Lenguaje Natural (PLN)....20
2.3.1 Introducción....20
2.3.2 Segmentación....20
2.3.3 Análisis Morfológico....21
2.3.4 Detección de Entidades....23
2.3.5 Etiquetado....24
2.3.6 Análisis Sintáctico....25
2.4 Análisis de la pregunta....25
2.4.1 Introducción....25
2.4.2 El tipo de la pregunta....26
2.4.3 Detección del foco....28
2.4.4 Expansión de la consulta....29
2.5 Búsqueda de información y extracción de la respuesta....30
2.6 RDF....31
2.6.1 Origen del RDF....33
2.6.2 Estructura de RDF....33
2.6.3 Búsquedas sobre RDF....34
2.7 SPARQL....35
2.7.1 Introducción....35
2.7.2 Componentes....36
2.7.3 Sintaxis....37
2.7.4 Funcionamiento....39
3. Diseño del sistema....41
3.1 Introducción....41
3.2 Sistema de QA....42

3.2.1 Bloque 1: Análisis de la pregunta....44
3.2.2 Bloque 2: Búsqueda de información....49
3.2.3 Bloque 3: Extracción de la respuesta....57
3.2.4 Bloque 4: Interfaz....60
4. Implementación....61
4.1 Desarrollo de la aplicación....61
4.2 Base de datos....61
4.2.1 ARC2....62
4.3 Implementación del sistema de QA....63
4.3.1 Bloque 1: Análisis de la pregunta....65
4.3.2 Bloque 2: Búsqueda de información....67
4.3.3 Bloque 3: Extracción de la respuesta....68
5. Pruebas....71
5.1 Introducción....71
5.2 Extractor de Wikipedia....71
5.2.1 Extracción de datos....71
5.2.2 Estructura de las fichas....74
5.2.3 Almacenamiento....76
5.2.4 Ficheros de configuración....78
5.3 Batería de pruebas....79
5.4 Respuestas....80
5.5 Interfaz de la aplicación....81
5.6 Alcance de la aplicación....81
5.7 Evaluación....83
5.7.1 Preguntas elementales....84
5.7.2 Preguntas encadenadas....86
6. Conclusiones y líneas de trabajo futuras....88
6.1 Conclusiones....88
6.2 Líneas de trabajo futuras....89
Bibliografía:....91

Índice de ilustraciones

Ilustración 1: Estructura típica de un sistema de QA 17
Ilustración 2: Esquema de procesado....20
Ilustración 3: Ejemplo de un análisis morfológico con la herramienta STILUS 21
Ilustración 4: Ejemplo de un análisis sin desambiguación con la herramienta STILUS 22
Ilustración 5: Ejemplo de un análisis con desambiguación con la herramienta STILUS....23
Ilustración 6: Ejemplo de distinción de fechas y nombres propios con la herramienta STILUS 24
Ilustración 7: Procesos de la consulta....26
Ilustración 8: Búsqueda típica con índices de términos 31
Ilustración 9: Ejemplo de grafo RDF 34
Ilustración 10: Sintaxis del operador SELECT 37
Ilustración 11: Sintaxis del operador CONSTRUCT....38
Ilustración 12: Sintaxis del operador ASK 38
Ilustración 13: Sintaxis del operador DESCRIBE....38
Ilustración 14: Modelado en forma de grafo de una estructura RDF 39
Ilustración 15: Ejemplo de consulta SPARQL 39
Ilustración 16: Esquema de un sistema de QA 42
Ilustración 17: Flujo de identificación de multipalabras....45
Ilustración 18: Algoritmo de análisis del etiquetado 47
Ilustración 19: Algoritmo de división de la pregunta....49
Ilustración 20: Algoritmo de la búsqueda de información....49
Ilustración 21: Ejecución de la extracción de la respuesta....58
Ilustración 22: Algoritmo de funcionamiento de la aplicación....59
Ilustración 23: Ejemplo de acierto....80
Ilustración 24: Ejemplo de respuesta aproximada 80
Ilustración 25: Ejemplo de pregunta no contestada 81
Ilustración 26: Ejemplo de pregunta mal contestada 81
Ilustración 27: Ejemplo de pregunta simple 82
Ilustración 28: Ejemplo de pregunta encadenada doble 82
Ilustración 29: Ejemplo de pregunta encadenada triple 83
Ilustración 30: Ejemplo de pregunta listado 83

Índice de tablas

Tabla 1: Homonimia (I).....10
Tabla 2: Homonimia (II).....10
Tabla 3: Palabras homógrafas.....11
Tabla 4: Palabras homófonas.....11
Tabla 5: Polisemia.....12
Tabla 6: Hipónimos.....13
Tabla 7: Cohipónimos.....13
Tabla 8: Problemas de cada disciplina.....15
Tabla 9: Listado de preguntas elementales.....86
Tabla 10: Listado de preguntas encadenadas.....87
Tabla 11: Resumen de resultados.....87

1. Introducción

1.1 Marco del proyecto

El proyecto se encuentra enmarcado en el área de la Inteligencia Artificial (IA), rama de la ciencia informática de la que actualmente dependen una gran cantidad de productos cotidianos, no solamente informáticos, sino también en el área de las grandes industrias donde han suplantando a los trabajadores para evitar posibles fallos humanos, o en los denominados edificios inteligentes y su correspondiente campo de la domótica, donde se pretende utilizar sistemas inteligentes para aumentar el confort, la seguridad o el ahorro energético, entre otros.

Debido a la informatización a pasos agigantados que está experimentando la sociedad actual, la IA está cobrando un gran protagonismo en la informática, debido a sus características entras las que se encuentra la automatización del trabajo, gracias a la cual se facilitan las tareas informáticas y por tanto de las tareas cotidianas.

Una de las áreas de la IA es la llamada Ingeniería Lingüística, basada en aplicar los conocimientos sobre el lenguaje natural, para reconocer, comprender, interpretar y generar lenguaje humano en todas sus formas.

Incluido en esta disciplina, hay un catálogo variopinto de utilidades como traductores automáticos, lematizadores, gestores de terminología...

Un nivel inferior, abarcado por este campo, se encuentra la Recuperación de Información (*information retrieval),* que consiste en dada una necesidad de información del usuario, localizar e interpretar los recursos de información pertinentes más relevantes para la resolución del problema planteado.

Los sistemas de respuesta automática (*QA – Question Answering,* en inglés), son un tipo especial de sistemas de Recuperación de Información en los que el sistema no devuelve una lista de documentos sino directamente la respuesta a la pregunta planteada, por ejemplo: *¿Cuál es la capital de Francia?*

Se incluyen en esta categoría, ya que para poder resolver el problema (la pregunta realizada), ahondan en los datos de los que disponen para obtener la mejor solución posible. De este modo, estos sistemas deben de tener unos documentos relacionados o incluso una base de datos, en donde puedan realizar esta búsqueda exhaustiva tratando de encontrar una respuesta concreta y específica, que le sea válida al usuario.

En la actualidad, los sistemas de QA cada vez están tomando más importancia, intentando convertirse en el futuro de los buscadores, de modo que el usuario no tenga que acceder a las páginas que se le ofrecen, sino

que directamente se le muestre la información que ha solicitado al buscador, ya que en la mayoría de los casos se está intentando encontrar una determinada respuesta

1.2 Objetivos

El objetivo principal de este proyecto consiste en el diseño y desarrollo de un sistema que sea capaz de “entender” preguntas en lenguaje natural (en este caso, español) y devolver una respuesta correcta al usuario, buscando para ello en una base de datos de información semántica.

Para llevar a cabo esta tarea se realizará un estudio y análisis de las técnicas actuales de los sistemas de QA, para elegir de entre ellas la que más se adapte al tipo de preguntas en las que se va a centrar este sistema.

En este caso, el sistema se basará en una base de datos de información semántica, pudiendo por tanto dar respuesta exclusivamente a preguntas cuya información esté contenida en la base de datos. La codificación y almacenamiento de la información se decidirá en el diseño del sistema.

El objetivo real es diseñar un algoritmo versátil y eficiente que permita traducir una consulta en lenguaje natural a una expresión de consulta en un lenguaje de base de datos que permita obtener la información requerida. Para ello se usarán diccionarios de traducción específicos para esta tarea.

Además del estudio y análisis de las técnicas se realizará una evaluación de aciertos en las respuestas ofrecidas por el sistema en forma de porcentaje, además de una breve explicación descriptiva sobre cada método empleado.

La aplicación descrita en este proyecto, trata de dar un buen soporte a las preguntas introducidas por el usuario, aunque no cumpla con todas las características de los sistemas QA, ni responda a todas las preguntas que se le introduzcan, ya que parte de unos datos y si se requiere algún tipo de dato que no sea sobre estas áreas, el sistema devolverá una respuesta del tipo “No se ha encontrado información”.

Por otra parte, y referente a la funcionalidad e implementación del sistema se cumplirán los siguientes requisitos:

- Partiendo de preguntas bien formadas (sin errores ortográficos y sintaxis sencilla) el sistema tendrá un nivel de aciertos aceptable.
- El sistema estará centrado en el español, aunque en la medida de lo posible será multilingüe
- El sistema será versátil, modular y portable, para posibles incorporaciones futuras en proyectos más grandes, o incluso en una interfaz Web, donde

tendría un entorno más “amigable”.

- La base de datos, de la que toma la información, será abierta para poder añadir entidades fácilmente, sin que ello propicie problemas en las consultas o en tiempo requerido para cada cuestión, y hará uso de estándares.
- El tiempo que el sistema tarde en buscar los datos y devolver una respuesta correcta será válido para una aplicación en tiempo real.

1.3 Contenido de la memoria

Este documento se encuentra dividido en los siguientes apartados:

Capítulo 1: El capítulo actual; se muestra una visión global sobre el proyecto en sí, así como una visión desglosada sobre los objetivos que se pretenden lograr con este sistema.

Capítulo 2: En este apartado, se expondrá la situación actual de los principales conceptos relacionados con la funcionalidad de la aplicación. Se comenzará explicando qué es la ingeniería lingüística, y las premisas de las que parte, acercando al lector términos que tendrán que ver a la hora de resolver los problemas relacionados con el lenguaje. Se continuará presentando qué es un sistema de respuesta automática, sus características, como funciona y algunos ejemplos de los actuales.

Capítulo 3: Este capítulo se corresponde con el diseño, por lo que se comentará cómo se ha creado la aplicación, los pasos que sigue hasta que se genera una respuesta, así como las llamadas a elementos externos al sistema.

Capítulo 4: Brevemente se dará una idea general de cómo funciona la herramienta utilizada (ARC), y los puntos generales de conexión con la base de datos y algunos algoritmos que el sistema utiliza para el correcto entendimiento de la pregunta formulada.

Capítulo 5: El desarrollo de este capítulo consiste principalmente en la interpretación de los resultados obtenidos durante la ejecución del sistema mientras fue sometido a un grupo de baterías de pruebas, con las que comprobar si se han cumplido los objetivos, descubrir las áreas de fallo y los motivos.

Capítulo 6: Tras el trabajo realizado, se ha llegado a unas conclusiones que son las aquí expuestas, además de posibles formas de mejorar el producto y nuevas líneas de investigación para ampliar su funcionalidad.

2. Estado del arte

En este apartado se muestra una visión global de los conceptos en los que se fundamenta este proyecto, describiendo y exponiendo la funcionalidad de los más relevantes así como numerosos ejemplos de aplicación y descripciones detalladas de sus partes.

Los sistemas de QA se basan principalmente en la Ingeniería Lingüística y en el Procesamiento del Lenguaje Natural, que serán comentados en el siguiente apartado de manera extensa, para comprender el significado y los principales problemas que afrontan estos sistemas.

Además, dado que el proyecto se enmarca en el campo de la Web Semántica se hablará brevemente de este campo, y por consiguiente del lenguaje que recomienda utilizar la W3C para la descripción de datos, el RDF.

2.1 Ingeniería lingüística

La Ingeniería Lingüística es una tecnología que utiliza nuestros conocimientos del lenguaje natural, es decir, aquel que utilizamos los humanos para comunicarnos entre nosotros, para mejorar la utilización de los sistemas informáticos, de manera que [1]:

- perfecciona nuestra interacción con ellos, consiguiendo una interacción hombre-máquina más adecuada (*HCI, Human Computer Interaction).*
- asimila, analiza, selecciona, utiliza y presenta la información de manera más eficaz.

Los sistemas de respuesta automática son aplicaciones dentro de la rama de la Ingeniería Lingüística, que se basan en dicha tecnología para cumplir sus objetivos.

2.1.1 Definición

La Ingeniería Lingüística como define la Wikipedia [2] *"es una disciplina también denominada Informática aplicada a la Lingüística e incluso Tecnología del Lenguaje, que, a su vez, tiene carácter multidisciplinar. La ingeniería lingüística se aprovecha del conocimiento desarrollado en el marco informático del procesamiento del lenguaje natural y del marco lingüístico nutrido por las disciplinas de la traducción, de la terminología y de la lingüística computacional, tanto en sus vertientes teóricas como aplicadas"*.

Expresado de manera simple se puede decir que es la tecnología del lenguaje, su procesamiento y todo lo que tiene que ver con él, bien sea traducciones, terminologías u otras aplicaciones.

2.1.2 El lenguaje y las necesidades de la ingeniería lingüística

El lenguaje es el medio principal a través del cual el ser humano intercambia información, expresa sentimientos o emociones, transmite conocimientos y crea, en definitiva, la cultura que se va transmitiendo de generación en generación. Ligado a este término desde el inicio de la civilización siempre han aparecido "*sociedad"* e *"información",* con lo que entra en escena un nuevo concepto "la sociedad de la información", donde destaca la entrada de las nuevas tecnologías con las cuales es mucho más sencilla la difusión del conocimiento.

Las nuevas tecnologías ofrecen al alcance de cualquiera una gran variedad de datos y de modos de interacción: en Internet se puede comprar, establecer nuevas relaciones, conversar con personas que están al otro lado del planeta, acceder a todo tipo de información o incluso asistir a sistemas de enseñanzas guiados por ordenador.

Con la aparición de Internet en el campo de las telecomunicaciones, el concepto de lo que se considera información cambia: no está mejor informado quien tiene más datos, sino quien dispone de los mejores medios para obtener exclusivamente los que necesita.

Este nuevo entorno comunicativo derivado de la aplicación de estas nuevas tecnologías plantea nuevos problemas: el acceso a la información debe ser eficaz, rápido y sencillo, y hay que prever la posibilidad de que los usuarios cometan errores.

Se puede disponer de un gran almacén de información, la propia Web es una fuente, pero si no se utilizan los sistemas apropiados de recuperación de información puede ser inservible por muchos conocimientos que se posean, ya que se ha de saber extraer lo adecuado y de manera eficiente.

De este modo, el desarrollo en las aplicaciones tendrá que destinar cierto esfuerzo en crear programas que procesen el lenguaje y el habla del amplio abanico de usuarios que pueden acceder, generando algoritmos y funcionalidades basadas en una gran cantidad de información.

Este gran problema llamado recuperación de datos, impone la tenencia de recursos de ingeniería lingüística capaces de desarrollar aplicaciones efectivas que cumplan estas expectativas, ya que como se ha mencionado antes, la parte más importante es que sean capaces de interpretar lo que se tiene almacenado o al alcance [3].

2.1.3 Procesamiento del lenguaje natural

Dentro de la ingeniería lingüística, en general existe una rama llamada procesamiento del lenguaje natural, que se encarga de estudiar el "enlace" que existe o debe existir entre los usuarios y el intérprete al que se le solicita la información, de tal modo que aunque no "hablen el mismo idioma" lleguen a entenderse.

El Procesamiento del Lenguaje Natural o *Natural Language Processing* (PLN o NLP), tiene por objetivo procesar y traducir automáticamente el lenguaje de los humanos a uno que las máquinas puedan analizar, interpretar y así recuperar la información eficazmente. Es decir el procesamiento del lenguaje natural para la recuperación de información trata de construir sistemas y mecanismos que permitan la comunicación entre personas y máquinas por medio de lenguajes naturales facilitando la búsqueda de información [4].

El lenguaje natural (LN), se convierte en la materia prima de dichos sistemas que deberán ser capaces de procesarlo, es decir, los elementos de la lengua, como fonemas, morfemas, palabras, oraciones, textos, así como las reglas que rigen el funcionamiento de la misma, para conseguir un nivel de comprensión por parte de las máquinas muy alto para que la recuperación intente ser siempre correcta.

Por tanto, hay que tener en cuenta las siguientes disciplinas lingüísticas en las que se apoya el lenguaje natural, y los que se van abordando a lo largo del análisis de la pregunta; comenzando por el inferior (fonética) hasta llegar al superior (pragmática).

- **Fonología**
- **Morfología**
- **Sintaxis**
- **Semántica**
- **Pragmática**

Fonética/Fonología

La Fonética [5] es una rama de la lingüística que se encarga de estudiar la producción, naturaleza física y percepción de los sonidos de una lengua, mientras que la fonología, describe el modo en que funcionan los sonidos de una manera más abstracta. Por lo tanto, este estudio se orientará más hacia la fonología que es donde más se ajusta.

Fonología

La fonología [6] es el estudio de los sonidos en el discurso, es decir, de los fonemas que son las unidades mínimas distintivas.

En este primer nivel, se analizan los sonidos del habla, es decir, su forma de realizarlos de manera física, aquello que la gente común define como "letras". Aunque dependiendo de la lengua, puede existir una gran variedad de sonidos semejantes a esa letra, es probable que realmente sólo existan unas pocas posibilidades fonológicas sobre dicho carácter.

Para analizarlo y por tanto sintetizar estos sonidos se han de organizar de algún modo, para ello los fonemas que existen en cada lengua (que suelen estar divididos a su vez en vocales y consonantes), se ilustran como pares de palabras que entre sí se diferencian tan sólo por un cambio mínimo, y son llamados pares mínimos.

Este análisis se basa en si existe un cambio de significado al realizar una leve permutación de sonidos. Estos cambios pueden ser sobre la misma letra, y tan solo cambiando lo que venga detrás. La letra “n” se pronuncia distinta si lo que le sigue es una palabra que comienza por vocal o consonante (*un ave/un golpe*).

La fonología, que a primera vista no debería tener mucha importancia en aplicaciones informáticas, es crucial a la hora de crear “ortografías” que se encarguen de intentar modelar la lengua en la que se vaya a trabajar, bien sea, para interpretar audio/vídeo que le llegue al programa, programas de escritura predictiva, programas que generen audio a partir de texto… En general cualquier aplicación que tenga que relacionar las palabras con los sonidos que representan.

Este nivel de lenguaje es donde más problemas se suelen dar ya que hay mucha ambigüedad, debido a las entonaciones, letras mudas, o fonemas similares para palabras totalmente distintas.

Morfología

La morfología se ocupa de analizar la constitución de una unidad menor que es la palabra. Se la ha considerado también como el estudio orientado a describir las partes constituyentes de las palabras; otra definición se inclina por establecer modelos de formación de palabras atendiendo a sus rasgos gramaticales; una tercera vía pretende describir la estructura de la palabra desde los mecanismos de formación de la palabra.

La determinación de las formas constituyentes se basa en su aspecto fónico, relacionando morfología con fonología. Los morfemas flexivos comportan relaciones con otros elementos de la cadena sintáctica.

"La morfología es entendida por un amplio abanico de trabajos como el estudio de la estructura de la palabra, por tanto, centra su atención preferentemente sobre los elementos formantes de una palabra, las relaciones entre los mismos, y las propiedades que derivan de su articulación en un resultado final que es la palabra" [7].

La palabra está compuesta por lexemas (raíz de la palabra) o morfemas (prefijos, sufijos o desinencias).

Si se observan los pares *irreal-real, incapaz-capaz, iletrado-letrado, inmortal-mortal*, se observan que los primeros términos de cada par denotan una negación de los segundos, usando formas fónicas parcialmente distintas. En el par *inmortal-mortal* su sentido es distinto, por lo que se determina una forma distinta homófona.

El lexema representa el conjunto de formas de palabras caracterizadas por ser manifestación de un mismo elemento léxico, y que, en cierto modo, vienen a representar en el plano significativo un mismo significado. Así *canto, canté, cantante, canción, cantata*... son manifestaciones del lexema o elemento léxico CANTAR y *niño, niña, niñas, niños* lo son de NIÑO.

Entre un conjunto de palabras pertenecientes a un mismo lexema se denomina raíz, al conjunto de palabras irreducible gramaticalmente.

La morfología se utiliza en programas informáticos, para descomponer y etiquetar en cada una de las partes en las que está subdividida la palabra, de ese modo, se puede deducir un análisis de la palabra, correcciones gramaticales y cualquier otra que se base en el tipo de palabras que componen las oraciones. O también, con una base de datos adecuada o conexión con la información necesaria se podrían crear aplicaciones que de algún modo utilicen los significados de los temas o bases.

Sintaxis

Se trata de la parte de la gramática que estudia la coordinación y unión de palabras para formar las oraciones y expresar conceptos [8]. En la informática, la sintaxis es el conjunto de reglas que definen las secuencias correctas de los elementos de un lenguaje de programación.

Como una subdisciplina perteneciente al campo de la lingüística, la sintaxis se encarga del estudio de las reglas que gobiernan la combinatoria de constituyentes y la formación de unidades superiores a éstos, como los sintagmas y oraciones.

Las formas más pequeñas en las que una forma más amplia se puede analizar son sus constituyentes sintácticos, una palabra o secuencia de

palabras que funciona en conjunto como una unidad dentro de la estructura jerárquica de una oración.

El paradigma actual de la ciencia refiere a la gramática generativa, que se centra en el análisis de la sintaxis como constituyente primitiva y fundamental del lenguaje natural.

Por otra parte, cabe destacar que el análisis sintáctico de una frase supone la búsqueda del verbo conjugado dentro de la oración, para distinguir entre el sintagma sujeto y el sintagma predicado**.** Para esto, una vez que se ubica el verbo, se pregunta quién realiza la acción. La respuesta constituye el sujeto, mientras que el resto es el predicado.

Por lo tanto, la sintaxis es muy utilizada en el análisis de la pregunta de cualquier sistema QA, ya que de este modo se puede reformular la pregunta y así extraer tanto el foco como la cuestión en sí, como se verá en el siguiente capítulo.

Semántica

Semántica [9] (del griego *semantikos***,** "lo que tiene significado"), es el estudio del significado de los signos lingüísticos (palabras, expresiones y oraciones)

Para ello se tienen que estudiar qué significa para los hablantes, cómo los designan, y también cómo los interpretan los oyentes.

Todo signo tiene dos vertientes: el significante o parte material del mismo y el significado o imagen que mentalmente genera el significante. Además hay que distinguir otro elemento que es el referente o elemento real, al que se refieren tanto significado como significante. No es lo mismo la palabra que designa un referente que el referente mismo.

Componentes del significado

El significado o imagen mental está compuesto por una serie de rasgos que todos los hablantes de una lengua asocian de una manera general a un significante. No obstante, este significado tiene dos componentes:

- **Denotación**. Son los rasgos objetivos. Es el significado concreto de una palabra fuera de contexto. Constituyen el núcleo semántico fundamental. Comunes para todos los hablantes. Es el significado que se puede encontrar en el diccionario.

- **Connotación**. Son los rasgos subjetivos. Son las significaciones que lleva añadidas una palabra de manera subjetiva. De modo que dependiendo de los hablantes, época o lugar no cobre un

significado distinto. Como por ejemplo, “manejar” en el idioma castellano y en el español de Sudamérica.

La semántica estudia las diferentes relaciones existentes entre un signo y todos los demás dentro de un contexto llamado léxico, que es el que constituye un sistema que permite a los hablantes la interpretación, conocimiento, adquisición y uso de ese léxico.

Relaciones entre significantes: la homonimia

- **La homonimia**

Se dice que dos palabras son homónimas si su significante es el mismo**;** es decir, están compuestas por los mismos fonemas, o su realización fonética coincide. No se trata de relación entre significados.

La relación homonímica más habitual se produce entre palabras de distinta categoría gramatical como se puede ver en la tabla siguiente.

Homonimia (I)	
Vino	Sustantivo común masculino, singular
Vino	Tercera persona del singular del pretérito indefinido del verbo venir

Tabla 1: Homonimia (I)

Pero también se produce en palabras de la misma categoría**.** Se da en aquellos casos en que el significado de las palabras no tiene ninguna relación, porque proceden de étimos distintos.

Homonimia (II)	
Hinojo	Planta medicinal
Hinojo	Rodilla

Tabla 2: Homonimia (II)

Dentro del concepto general de homonimia, se pueden distinguir varios casos mostrados a continuación.

Palabras homógrafas: Tienen las mismas grafías y los mismos sonidos

Palabra homógrafas	
Haya	Árbol
Haya	1ª y 3ª pers. Sing. presente del subjuntivo del verbo haber

Tabla 3: Palabras homógrafas

Palabras homófonas: Tienen los mismos sonidos, pero distinta grafía.

Palabras homófonas	
Aya	Criada
Halla	2ª pers. Sing. imperativo del verbo hallar

Tabla 4: Palabras homófonas

Todas ellas son homónimas. Las dos primeras son homógrafas, y las dos últimas son homófonas, entre sí, y respecto a las anteriores.

Relaciones entre significado y significante: monosemia, polisemia y sinonimia

- **Monosemia**

  Es la relación habitual que existe entre el significado y el significante en una palabra. A un significante se corresponde un solo significado.

  Por ejemplo, la palabra *bolígrafo* expresa un referente que sólo puede ser evocado mediante ese significante.

- **Polisemia**

  Una palabra es polisémica cuando se puede expresar con ella varios significados. O, dicho de otra forma: un significante puede tener varios significados.

  La polisemia se distingue de la homonimia en que se trata de una relación entre los dos planos del signo lingüístico: los diferentes significados de una palabra tienen, o han tenido, un origen común.

Polisemia	
Araña	Animal artrópodo o lámpara
Gato	Animal Mamífero o herramienta

Tabla 5: Polisemia

La polisemia es uno de los mecanismos más eficaces de economía lingüística, pues permite expresar varios significados con un único significante

.

- **Sinonimia**

  Dos o más palabras son sinónimas si tienen el mismo significado. Es decir, la sinonimia consiste en la igualdad de significado, cuando existen diferentes significantes.

  Se pueden distinguir diversas formas en que puede presentarse la sinonimia:

  Sinonimia conceptual: Los significados denotativos son plenamente coincidentes. Ej: *listo=inteligente*

  Sinonimia connotativa: Puede, en ocasiones, no haber coincidencia denotativa; sin embargo, esto no impediría que se consideren sinónimos por los valores connotativos que encierran. Ej: *listo=zorro*

  Sinonimia contextual: En determinados contextos, se pueden establecer ciertas sinonimias que serían impensables en otros. Ej: *listo=preparado*, en contextos como *¿Estás listo?*

Relaciones entre significados: el campo semántico

Hiperonimia e hiponimia

Se llama hiperónimo a la palabra cuyo significado abarca al de otras, que se conocen como hipónimos. Los hipónimos a los que se refiere una palabra son, entre sí, cohipónimos.

Se pueden distinguir las relaciones siguientes ilustradas con ejemplos en tablas.

- **Relaciones de inclusión:** Un conjunto de palabras puede estar englobado dentro de otra palabra que las incluya a todas.

Hiperónimo	Hipónimos	
Flor	Clavel	Hipónimos
	Rosa	
	Marga rita	

Tabla 6: Hipónimos

- **Relaciones lineales.** En otros casos, se establecen relaciones de sucesión. Así sucede, con los nombres de los meses o los días: *Enero, Febrero,..., Diciembre; lunes, martes,..., domingo.*

Hiperónimo	Hipónimos	
Día	lunes	Cohipónimos
	martes	
	miércoles	

Tabla 7: Cohipónimos

Valores expresivos del significado

El significado puede convertirse en un elemento de máxima efectividad expresiva. Si se tienen en cuenta los elementos de la comunicación, la situación comunicativa va aclarar el significado de muchas palabras. Pero a veces, el contexto referencial hará que surjan significados nuevos, que antes no estaban presentes.

Hay que tener en cuenta que toda palabra tiene un significado denotativo y un significado connotativo. Las connotaciones pueden ser positivas o negativas, siempre dependiendo del hablante que las considere.

Palabras tabú son aquellas que no se pronuncian, porque tienen una carga connotativa despectiva. Se sustituyen por otras palabras que designan la misma realidad, pero sin esas connotaciones peyorativas. Son los denominados eufemismos (del griego: palabra bien sonante).

Ciertas partes del cuerpo siempre han estado sometidas a fenómenos de tabú. Pocos son los alumnos que se atreverían a pedir al profesor permiso para ir a *"mear"*. Y en el caso de que la hicieran se consideraría una falta de respeto. La normal es que se utilice el eufemismo: *"¿Puedo ir al servicio?".*

Al igual que existen eufemismos, también hay disfemismo**.** Cuando la palabra tabú se sustituye por otra, pero de carácter humorístico. Ej: En vez de *muerto,* se utiliza el disfemismo *fiambre.*

Pragmática

La Pragmática [10] es el estudio del modo en que el contexto influye en la interpretación del significado. El contexto debe entenderse como situación, ya que puede incluir cualquier aspecto extralingüístico.

Incluye en sus análisis los factores sociales, psicológicos, culturales, literarios, que determina la estructura de la comunicación verbal y sus consecuencias. En esta se relacionan la semántica y la sintaxis: la semántica hace abstracción de los usuarios y la sintaxis expresa la relación entre los signos sin tener en cuenta a los usuarios; sintetizando todo el proceso en el estudio del qué se dice y lo que literalmente se quiere decir.

Es fundamental analizar también las huellas que emisor y receptor dejan en el texto. Así, por ejemplo, la presencia de un YO que se dirige a un TÚ puede imprimir una cierta fuerza persuasoria al mensaje, al introducirse, consciente o inconscientemente, el autor en el texto en un intento de modificar la conducta de la persona que recibe el mensaje.

La pragmática busca analizar los cambios que se presentan en determinados contextos, porque estos contextos permean las palabras, los gestos y el mensaje en general de hablantes, y la o las interpretaciones hechas por los oyentes.

2.1.4 Problemas en el procesamiento del lenguaje natural

Al procesar el lenguaje natural en utilidades que requieran de dicha actividad, empiezan a surgir los problemas en los campos anteriormente vistos, algunos generando problemas irresolubles o de difícil solución. El problema que se da siempre es la ambigüedad, bien sea por su manera de escribirse (*haya,* árbol; *haya,* verbo haber) o de pronunciarse (*ola*, onda acuática; *hola*, saludo), como ya se vio en los apartados anteriores. La ambigüedad a su vez acarrea problemas que se van arrastrando a lo largo del análisis, tanto del morfológico como del sintáctico posteriormente. Con lo que es el principal problema al que se ha de enfrentar cualquier sistema de PLN.

En la siguiente tabla se muestra, a modo de resumen, los problemas que se generan por niveles de menor a mayor.

Disciplina	Problema
Fonología	Afecta a los programas que utilicen reconocedor de voz Problemas con las pautas de voz de los usuarios Ej: *Vaca / Baca* *Hora / Ora*
Morfología	Confusiones a la hora de realizar el análisis morfológico en palabras que se escriben igual (homógrafas) Ej: *"La casa"* → *verbo casar* → *edificio*
Sintaxis	Concordancia de género y número
Semántica	La homonimia y la polisemia
Pragmática	Desconocimiento del contexto Problemas con la persona Ej: "*¿Tienes hora?*

Tabla 8: Problemas de cada disciplina

2.2 Sistema de respuesta automática a preguntas

Los sistemas de Question Answering o sistemas de QA, son aplicaciones informáticas cuyo objetivo es la búsqueda de respuestas para preguntas que les son formuladas de una manera natural, es decir, sin tener en cuenta que van dirigidas a una máquina. Estos sistemas deben ser capaces de interpretar ese lenguaje y generar una respuesta correcta. De ese modo serían capaces de contestar cuestiones como: "*¿En qué año nació el presidente de Cuba?"* o *"¿Cuál es la población de la capital de España?"*

2.2.1 ¿Qué es la búsqueda de respuestas?

Más allá de aspectos filosóficos, la búsqueda de respuestas no es más que cubrir una necesidad, es decir, dar solución a un problema, y en este caso es

tan sencillo como responder.

El problema más importante para dar esta respuesta, es de dónde sacar la información necesaria para dar la correcta.

Es sencillo responder a una pregunta si se dispone de la información adecuada y, evidentemente, si no se dispone de esta información, es complicado responderla (por mucho que se invente o improvise la respuesta). Por esta misma razón se podría decir que casi no existen preguntas sin respuesta, sino, falta de información.

Típicamente, los usuarios de los ordenadores, realizan consultas en buscadores para obtener la información necesaria y satisfacer así sus preguntas. Primeramente obtienen una lista de enlaces y/o información y después, tras un tiempo analizando y seleccionando, obtienen la mejor (en la mayoría de los casos) de las respuestas, y por lo tanto, la información que buscaban.

Los sistemas de búsqueda de respuesta realizan esta búsqueda, análisis y selección por sí mismos evitando así al usuario estas tareas.

El problemas de los sistemas de respuesta automática es que al no tratarse de personas, la capacidad de razonamiento es limitada, simplemente utilizan las instrucciones que se les hayan introducido como razonamiento, y de ahí parte uno de los grandes problemas de estas aplicaciones: el emular a una persona como si ésta estuviera realizando una consulta, buscando la información y seleccionando la respuesta adecuada para cada caso.

2.2.2 Arquitectura de un sistema de respuesta automática

Comúnmente los sistemas de QA presentan una arquitectura del tipo especificado en la ilustración [11].

Ilustración 1: Estructura típica de un sistema de QA

- **Análisis de la Pregunta:** Esta parte se encarga de transformar una consulta a la forma correspondiente para que sea "inteligible" por la máquina. Para ello se utilizan herramientas de procesamiento del lenguaje natural que se encargan de interpretar la consulta: categorizándola, analizándola morfológicamente, desechando palabras sin significado, etc.…

  En definitiva, se extrae la suficiente cantidad de información para que el sistema sea capaz de interpretar la pregunta, y llegar a una solución con esta información extraída.

- **Búsqueda de Información:** Los sistemas de QA deben disponer de un sistema de Recuperación de Información, que devuelva una colección de documentos, enlaces, propiedades… que contengan la información solicitada. Esta información puede estar almacenada de distintas maneras, pero el sistema de Recuperación de Información debe seleccionar la importante y relevante que esté relacionada con la información solicitada, y la salida del análisis de la pregunta.

- **Extracción de la Respuesta:** Tras obtener una lista de fragmentos de texto relevante, susceptible de contener la respuesta a la pregunta formulada se ha de proceder a la elección de la mejor respuesta y una vez seleccionada, extraerla. Para elegir la mejor respuesta existen diferentes métodos dependiendo del tipo de información que se quiera devolver, o la cantidad.

  Por ejemplo: se pueden solicitar listas de objetos o un solo objeto, difiriendo en ese caso en la cantidad de respuestas.

2.2.3 Características de los sistemas QA

Estas características fueron fijadas en el 2002 por un grupo de investigadores, sobre como deberían ser los sistemas de QA, aunque no siempre cumplan con todas estas características [12]:

- **Exactitud:** El sistema debe ser todo lo exacto que pueda, y en caso de dudar no contestar; puesto que una pregunta mal contestada es peor que la ausencia de contestación.

- **"Usabilidad":** Ha de responder, en la medida de lo posible, con el mismo formato de la pregunta.

- **Independiente del tiempo:** Ha de dar respuesta tanto a preguntas actuales y recientes, como a preguntas relacionadas con la antigüedad.

- **Relevancia:** Debe dar respuestas relevantes y si no es posible debe consultar al usuario.

- **Respuesta Completa:** La respuesta ha de ser lo más completa posible aun siendo necesaria la unión de frases que quizás están en distintos documentos.

2.2.4 Desarrollo mundial

Hoy día existen diferentes grupos en todo el mundo que investigan sobre los sistemas de QA. En Estados Unidos, en 1992, nació el TREC [13] (*Text Retrieval Conference*) con el fin de crear una comunidad de investigadores en sistemas de recuperación de Información. En Europa existe el CLEF [14] (*Cross-Language Evaluation Forum*) con los mismos objetivos pero haciendo más hincapié en la diversidad de las lenguas.

Desde la creación de estos dos foros, los sistemas de QA mejoran año a año, el número de investigadores y desarrolladores aumenta y a la vez el número de países implicados. Gracias en gran medida a CLEF, se ha aumentado el interés de otros sistemas de QA como pueden ser los sistemas bilingües o multilingües.

Es evidente que estos sistemas requieren de técnicas de traducción y de otras medidas de procesamiento que no se van a tratar en el presente escrito.

Actualmente existen una gran variedad de sistemas de QA en funcionamiento, por citar algunos como ejemplo se podrían mencionar:

- START, Natural Language Question Answering System [15]

  START es un sistema de pregunta y respuesta basado en técnicas de Recuperación de Información, para la Web y desarrollado por el MIT (Massachusetts Institute of Technology), en 1993.

  El funcionamiento de START consiste en comprobar la correspondencia de su base de conocimiento con la cuestión que le ha sido introducida. Para ello, la pregunta se segmenta en forma de árbol, siendo cada uno de sus nodos una pregunta distinta.

  START también trabaja con una amplia variedad de formatos audiovisuales como imágenes, vídeos o archivos de sonido. La clave en la búsqueda de este tipo de archivos es una técnica llamada "*Natural Language Annotation*" que consiste en asociar unas palabras clave a este tipo de información.

  La mayor desventaja de este sistema es que solo funciona en la lengua inglesa.

- AnswerBus [16]

  AnswerBus es un proyecto de la Universidad de Saarbrücken de sistema de QA. Se basa en la Recuperación de Información a través de sentencias que pueden estar en inglés, alemán, francés, español, italiano y portugués. Las frases son buscadas en la Web mediante cinco buscadores y directorios como son: Google, Yahoo!, WiseNut, Altavista y Yahoo News, mediante los cuales se extraen páginas Web que contienen respuestas potenciales, que posteriormente serán analizadas y evaluadas mediante un sistema de puntuación.

  Es multilenguaje y tiene un interfaz online abierto a cualquier usuario.

- Answers.com [17]

  Servicio online que incluye varias funcionalidades como son WikiAnswers, donde son los usuarios al igual que en otros modelos "wiki" se encargan de formular y responder las preguntas, o ReferenceAnswers, donde se busca respuesta a las preguntas en varias fuentes de información fiables.

  Su motor de respuestas fue creado por Answers Corporation, y el soporte multilingüe fue introducido posteriormente a su creación.

- Ask Jeeves [18]

  También conocida como ask.com, es un motor de búsqueda de internet. En la actualidad cuenta con una versión diferente para los

países: Reino Unido, Italia, Alemania, España, Japón y Países Bajos; además de otras versiones para diferentes portales temáticos como su versión para niños.

Fue el 1er buscador comercial del tipo pregunta-respuesta para web. Soporta una amplia variedad de consultas en inglés y las búsquedas por palabras clave.

2.3 Procesamiento del Lenguaje Natural (PLN)

2.3.1 Introducción

Los sistemas de QA utilizan unas técnicas típicas de PLN, que a veces pueden diferir del modelo básico del que parten, pero que en esencia sirven de esquema para todos.
La siguiente figura muestra los pasos típicos que siguen los sistemas de QA

Ilustración 2: Esquema de procesado

2.3.2 Segmentación

Principalmente, la segmentación consiste en dividir el texto por frases, y estas frases a su vez en palabras. De este modo, una vez se tienen las palabras por separado, puedan ser categorizadas y se interprete el significado que aportan al conjunto del que son parte.

2.3.3 Análisis Morfológico

El análisis morfológico consiste en determinar la forma, clase o categoría gramatical de cada palabra de una oración. Un ejemplo es el mostrado en la figura siguiente (utilizando la herramienta STILUS) [19]:

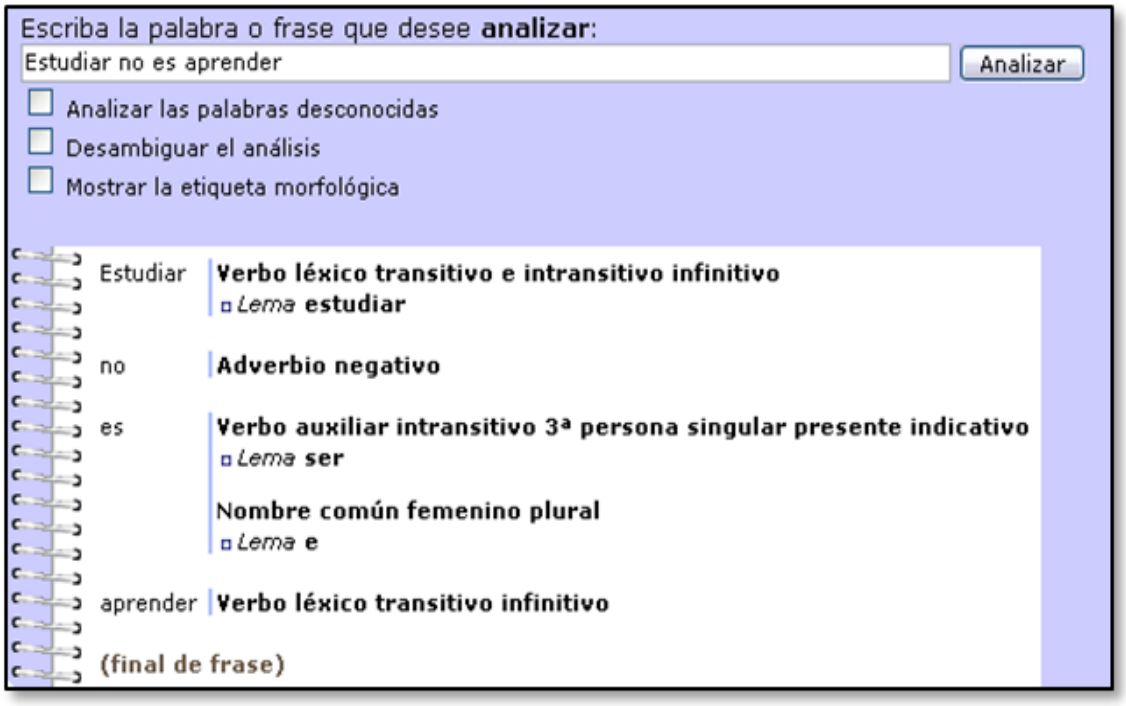


Ilustración 3: Ejemplo de un análisis morfológico con la herramienta STILUS

Como se aprecia en la ilustración, realizar el análisis morfológico realmente consiste en dar una etiqueta a las palabras, bien sea un código que interpretará posteriormente el programa o bien sean dándole los rasgos completos.

A la hora de realizar este etiquetado, surge un problema que después jugará un papel muy importante para realizar correctamente los análisis o no, la desambiguación.

En el caso de STILUS, el etiquetado consiste en asignar una marca (etiqueta) formada por cifras y letras en las que se indica la categoría gramatical, género, número, persona, tiempo, modo…

Esto ayuda a elegir las palabras que se van a buscar ya que es muy común dar más importancia a un nombre que a una preposición.

En los ejemplos que a continuación se muestran, se observa cómo tras una desambiguación el etiquetado de la palabra "*capital*" cambia. El sistema es capaz de diferenciar entre la referencia a una ciudad o a un valor monetario.

Estos cambios pueden afectar a palabras, con cambios tan mínimos como

puede ser la sustitución del artículo que lo precede por otro. Por ejemplo, en la frase *"La capital de Brasil es…",* el artículo "*la*" indica que el significado de *capital* se refiere a la ciudad que administra un país, mientras que si se sustituye el artículo previo a la palabra, quedando así la oración *"El capital de Brasil es..",* se le da a capital otro significado distinto. Esta comparativa aparece reflejada en las ilustraciones siguientes.

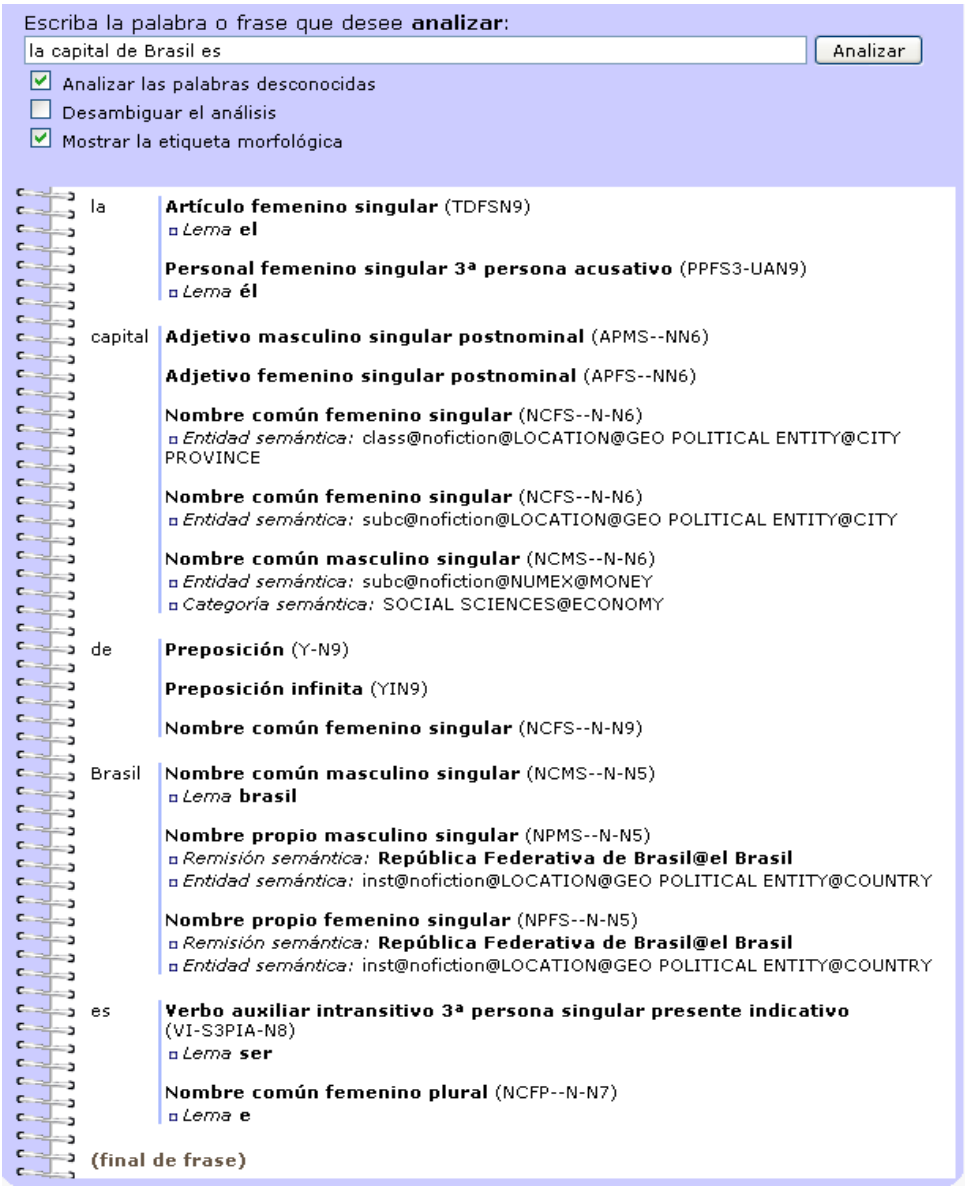


Ilustración 4: Ejemplo de un análisis sin desambiguación con la herramienta STILUS

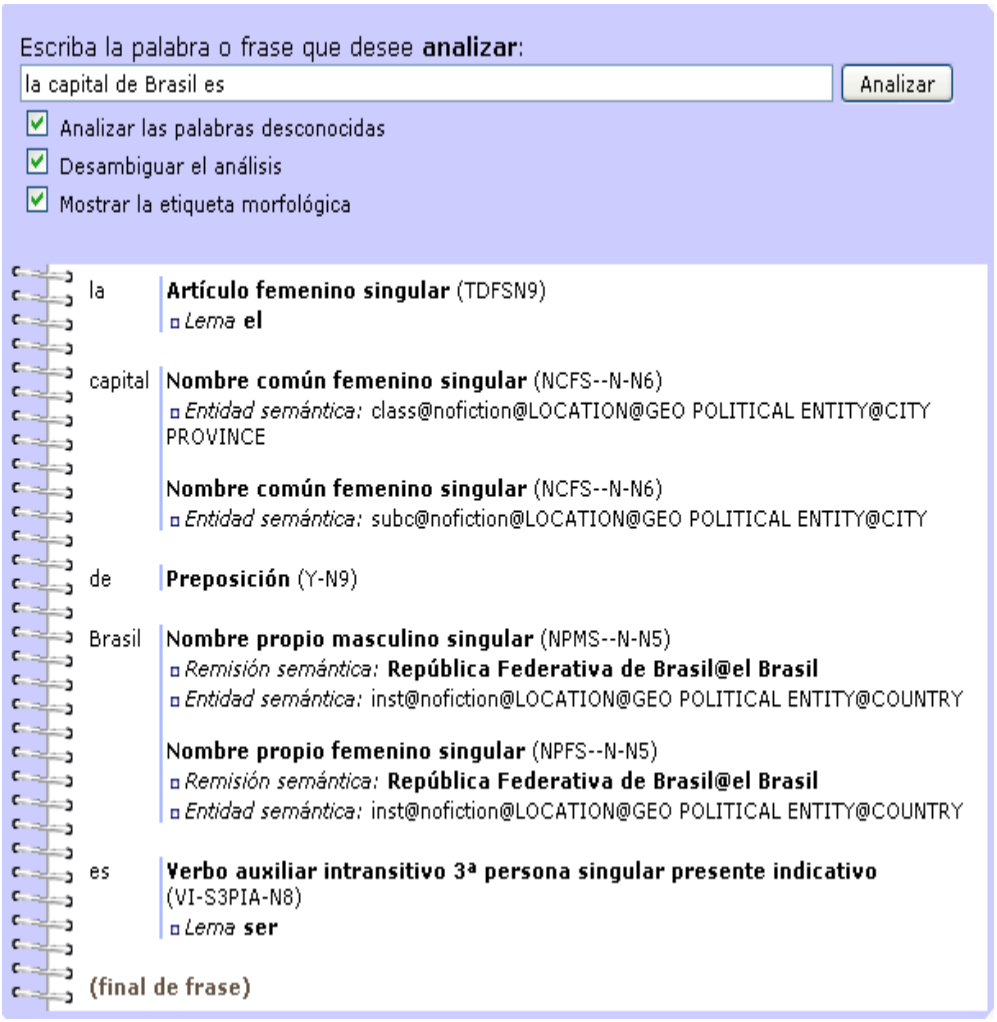


Ilustración 5: Ejemplo de un análisis con desambiguación con la herramienta STILUS

2.3.4 Detección de Entidades

Algunos sistemas y herramientas del procesamiento del lenguaje natural son capaces de detectar entidades tipo fechas, ciudades, personas, etc.

Estas detecciones permiten encontrar multipalabras que se refieren a un mismo objeto pero que se encuentra fraccionado, como las fechas como se aprecia en el ejemplo, o los nombres propios. Ejemplo: "*Rafael Nadal*" se debe referir a la misma persona, no "*Rafael*" por una parte al cantante y "*Nadal*" al tenista o jugador de fútbol.

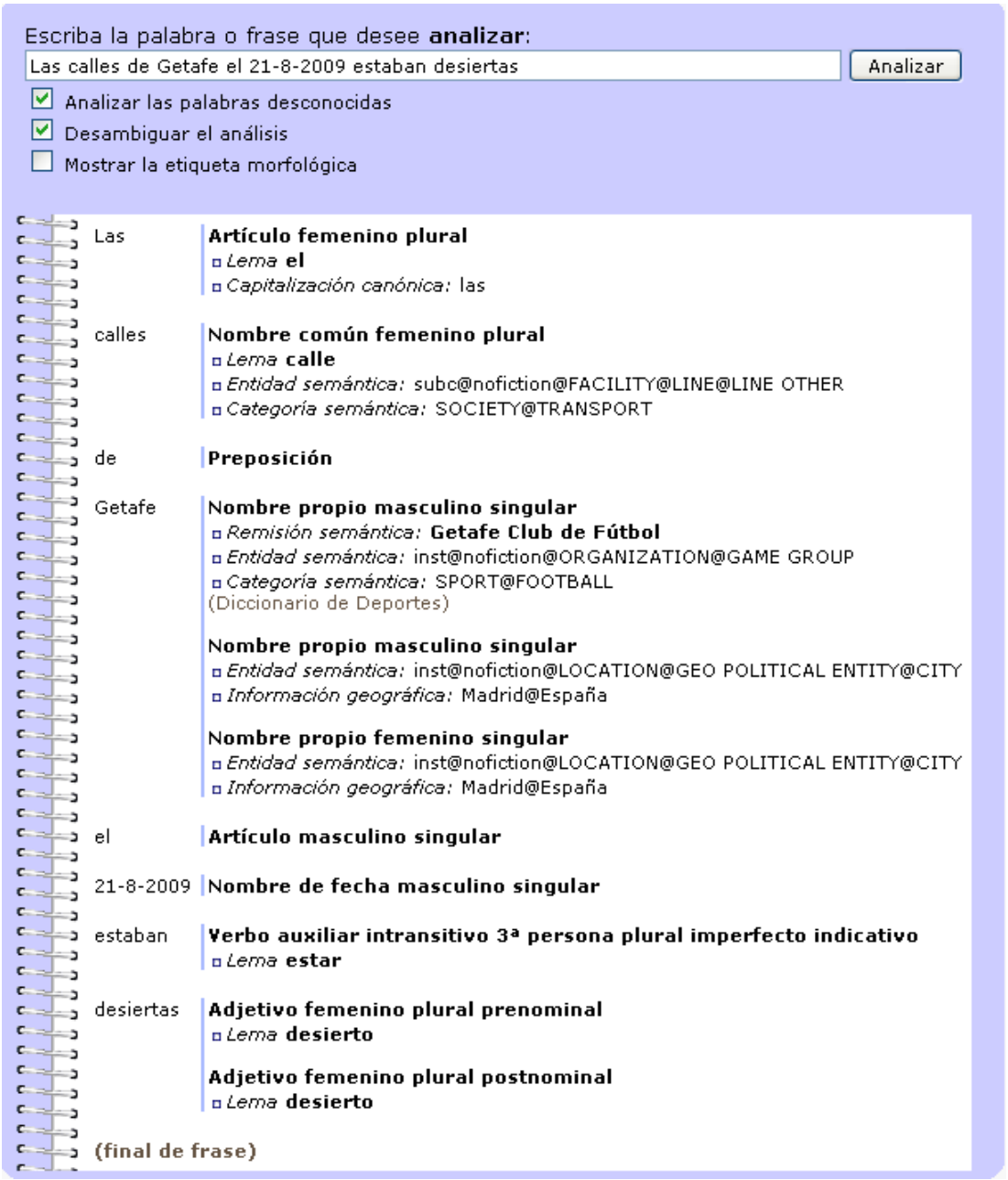


Ilustración 6: Ejemplo de distinción de fechas y nombres propios con la herramienta STILUS

2.3.5 Etiquetado

Este etiquetado se basa en lematizar cada una de las palabras en las que se ha fraccionado el texto. Lematizar consiste en la reducción de las diferentes formas flexivas de una palabra a la forma canónica, su lema o representante de toda la familia flexiva.

Por convenio esta reducción consiste en reagrupar las distintas inflexiones de un verbo en el infinitivo; el singular y el plural de un sustantivo en el

singular; el masculino y el femenino de un adjetivo en el masculino.
Con esto se consigue identificar familias de palabras para considerarlas como una sola. Es un dato muy importante para la búsqueda de información ya que se consigue hacer independiente la búsqueda de tiempos verbales y otros.

2.3.6 Análisis Sintáctico

El último paso es el análisis sintáctico. De todos los niveles de análisis, la sintaxis ha sido durante mucho tiempo y aún sigue siendo el nivel al que la lingüística le ha prestado mayor atención. Esta casi exclusiva atención se justifica por dos razones principales en cuanto al tratamiento automático del lenguaje natural:

1. El procesamiento semántico funciona sobre los constituyentes de la oración. Si no existe un paso de análisis sintáctico, el sistema semántico debe identificar sus propios constituyentes. Por otro lado, si se realiza un análisis sintáctico, se restringe enormemente el número de constituyentes a considerar por el semántico, mucho más complejo y menos fiable. El análisis sintáctico es mucho menos costoso computacionalmente hablando que el análisis semántico (que requiere inferencias importantes). Por tanto, la existencia de un análisis sintáctico conlleva un considerable ahorro de recursos y una disminución de la complejidad del sistema.

2. Aunque frecuentemente se puede extraer el significado de una oración sin usar hechos gramaticales, no siempre es posible hacerlo.
   La sintaxis contempla dos modos diferentes, pero no por ello opuestos, de análisis. El primero es el análisis de constituyentes o análisis de estructura de frase: la estructuración de las oraciones en sus partes constituyentes y la categorización de estas partes como nominales, verbales, adjetivales, etc. El segundo es el análisis de las relaciones o funciones gramaticales: la asignación de relacionales gramaticales tales como Sujeto, Objeto, etc.

2.4 Análisis de la pregunta

2.4.1 Introducción

El análisis de la pregunta es quizás la parte más importante del sistema. Lo que pretende este bloque es aumentar las posibilidades de encontrar la

respuesta, para lo cual se utilizan distintas metodologías. Para hacerse una idea de los procesos que sufre la consulta basta con echar un vistazo a la figura siguiente.

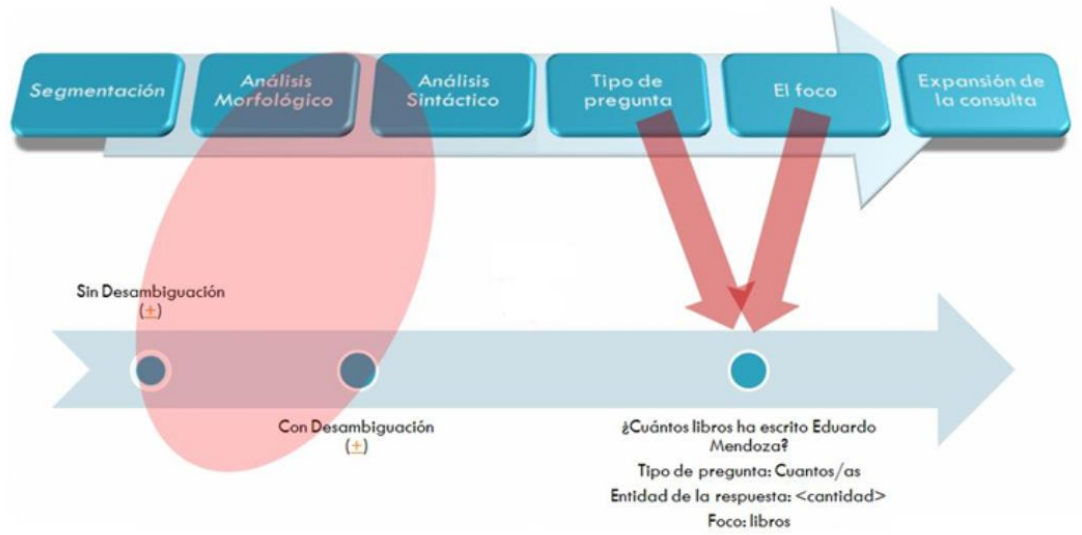


Ilustración 7: Procesos de la consulta

2.4.2 El tipo de la pregunta

Una buena clasificación de la pregunta ayudará a acotar el campo de búsqueda de la respuesta, centrando así la categoría semántica de la respuesta y ayudando, por lo tanto, a "intuir" la respuesta esperada.

Esta medida ayuda de manera significativa a la funcionalidad de los sistemas de respuesta automática, no obstante, la cantidad de tipos de preguntas puede llegar a ser muy elevada. Este problema originó el estudio del llamado "foco" del que se hablará más tarde.

Cabe distinguir los siguientes tipos de preguntas en los que se suele englobar cualquier pregunta:

- **Sí / No**

  Son aquellas preguntas que esperan una contestación de forma "booleana", sí/no, verdadero/falso... Suelen ser las más sencillas de tratar y su evaluación es prácticamente trivial. No son preguntas muy frecuentes en los usuarios.

  Ej:

  ¿Es Rajoy el actual presidente del gobierno?

- **Concretas**

  Lo que el usuario busca es un dato en concreto. Estas preguntas son relativamente fáciles de categorizar ya que, el usuario deja claro que quiere algo en concreto ayudando a esta categorización y a su vez, de manera implícita a la acotación del campo de búsqueda.
  Ej:
  ¿Cuál es el animal más veloz?

- **Listas**

  Típicamente, lo que el usuario quiere saber, es el resultado de la unión de un número de datos concretos:
  Ej:
  ¿Qué países forman parte de la OTAN?

  El usuario, claramente, está pidiendo una lista de países. Este tipo de preguntas suelen ser bastante problemáticas debido a que, el sistema, nunca puede saber la cardinalidad de la misma, no podría tampoco, saber si existen contradicciones.

- **Definiciones**

  Son preguntas normalmente sencillas de identificar y categorizar. Por contra, son bastante difíciles de responder. El cómo seleccionar una respuesta adecuada es quizás la parte más compleja de estas preguntas.
  Ej:
  ¿Qué es la vida?

  Los aspectos filosóficos, científicos, ideológicos, etc. así como el tipo de usuario (experto, casual, periodista, etc.) juegan un papel que para el sistema es imposible de comprobar. Por otra parte, la tarea de escoger una parte de un artículo como definición de algo, es más que compleja.

- **Preguntas Resumen**

  Típicamente la respuesta a estas preguntas está oculta en varios documentos, por ejemplo:

  ¿Qué se dijo sobre el avance del SIDA en el continente Africano, en la última Conferencia Mundial Contra el SIDA?

Para poder contestar a esta pregunta haría falta, posiblemente, obtener datos de varios artículos, o generar un resumen. Podría decirse, que respondiendo con una lista de datos (en este caso, frases sobre el VIH) el usuario obtendría su respuesta.

- **Preguntas de Contexto**

  Cuando dos personas entablan conversación, es frecuente que se hagan preguntas en las que interviene el contexto o la situación:
  Ej:
  ¿Quién es el ministro de exteriores?

  Es evidente que la respuesta varía con el contexto, en este caso depende del país. En este tipo de preguntas es necesario mantener el contexto para poder responder adecuadamente.

- **Preguntas en Temas Específicos**

  Cuando un grupo de personas entablan conversación sobre un tema, todas las preguntas obtienen un contexto implícito que un sistema de QA no puede obtener. Por ejemplo, si dos personas hablan sobre el grupo musical Los Beatles y la pregunta es:

  ¿Cuándo sacaron su último disco?

  Evidentemente un sistema de QA no puede dar una respuesta correcta a dicha pregunta, es más, ni si quiera es capaz de dar una respuesta, ya que no sabe de que se está hablando, es decir, no conoce el tema ni el contexto de la pregunta.

- **Preguntas Abiertas**

  Al igual que las definiciones, este tipo de preguntas son bastante complejas. Son preguntas tipo *¿cómo?, ¿por qué?*
  Ej:
  ¿Cómo realizar un estudio sobre Procesamiento de Lenguaje Natural?

  El sistema deberá recopilar listas de datos de diferentes documentos y generar un único documento automáticamente mediante técnicas de generación de textos.

2.4.3 Detección del foco

El foco es una palabra o conjunto de palabras que toma especial relevancia

dentro de un contexto y que está directamente relacionada con el tipo de respuesta. La siguiente pregunta se podría analizar:

¿Cuántos libros ha escrito Eduardo Mendoza?

- Tipo de pregunta: Cuántos/as
- Entidad de la respuesta: <cantidad>
- Foco: libros

Es evidente que la respuesta será algo del tipo <cantidad> <libros>, es decir, que aun teniendo el mismo tipo de pregunta, ahora se exige que esa cantidad sea de un tipo concreto (libros).

2.4.4 Expansión de la consulta

Uno de los problemas más importantes en estos sistemas es que es muy frecuente que las preguntas realizadas por el usuario no encajen exactamente con los documentos que se dispone por usar diferente terminología

Para solucionar este problema existen métodos de expansión de la consulta, a continuación son mencionados los actuales métodos.

2.4.4.1 Expansión por Sinónimos y palabras "vacías"

Como es lógico un programa no solamente puede esperar que el usuario introduzca la palabra exacta con la que se tiene definida en el sistema de información, el dato en cuestión.

Para intentar "entender" mejor la pregunta que se ha realizado se utiliza la expansión de sinónimos que se ejecuta sobre las palabras que no son vacías (es decir, que no aportan "ningún" significado a la frase). Con estas nuevas palabras relacionadas con las no-vacías hay más posibilidad de contestar adecuadamente a la pregunta, ya que en un buen programa que utilice análisis sintáctico podrá definir cada parte de la frase, y utilizar la acepción correspondiente en cada caso, puesto que dependiendo del contexto, una misma palabra puede tener significados totalmente distintos.

Las palabras vacías mencionadas anteriormente pueden variar mucho dependiendo de la funcionalidad implementada, así para un programa sin análisis sintáctico o morfológico, los determinantes como "el" o "una" podrían ser eliminados debido a que no contribuyen con ningún dato adicional; mientras que para otros sistemas que utilicen análisis sintáctico, una palabra vacía, como lo pueden los ser determinantes, puede ayudar a discernir en la desambiguación.

2.4.4.2 Reformulación de la pregunta

Otra de las técnicas empleadas en los sistemas de QA, es la llamada reformulación de la pregunta, que consiste en intentar "adaptar" la pregunta introducida por el usuario a unas plantillas determinadas que se tienen almacenadas, y que gracias a ellas, se consigue dar una orientación al programa sobre lo que realmente se busca. A continuación, se muestra un ejemplo de cómo funciona la reformulación.

- P: ¿Quién mató a Kennedy?
- R1: <alguien> mató a Kennedy.
- R2: <alguien> fue el asesino de Kennedy.
- R3: Kennedy fue asesinado por <alguien>.

2.5 Búsqueda de información y extracción de la respuesta

La búsqueda de información se ha de realizar tras haber analizado la pregunta y extraído los elementos útiles para la búsqueda y por lo tanto para una correcta, y lo más posible completa, respuesta.

Una vez se tienen estos elementos, se procede a buscar la información solicitada, y a intentar extraer una información que se corresponda.

Los sistemas de QA, dado el gran volumen de información que tienen que tratar, utilizan técnicas para acceder más rápido a los datos, y por consiguiente optimizar el tiempo de respuesta.

La más importante es el uso de índices sobre la información contenida. Con esta técnica se consigue mejorar notablemente el tiempo de respuesta, ya que al acceder por medio de índices a la información, evita tener que recorrer el grueso de la información. Normalmente además de indizar, se realiza un paso previo que consiste en procesar y normalizar los datos para que al ser procesados por la base de datos, quede una estructura simple y se generen unos índices lo menos complejos posibles, ya que a medida que la complejidad crece, la velocidad decrece.

Para la aplicación descrita, este procesamiento no se realiza ya trae la información, simplemente se introduce en la base de datos, ya que al estar en formato RDF (que se verá descrito en el siguiente capítulo), no es necesario un procesamiento, puesto que ya se encuentran estructurados.
En la ilustración se muestra el funcionamiento.

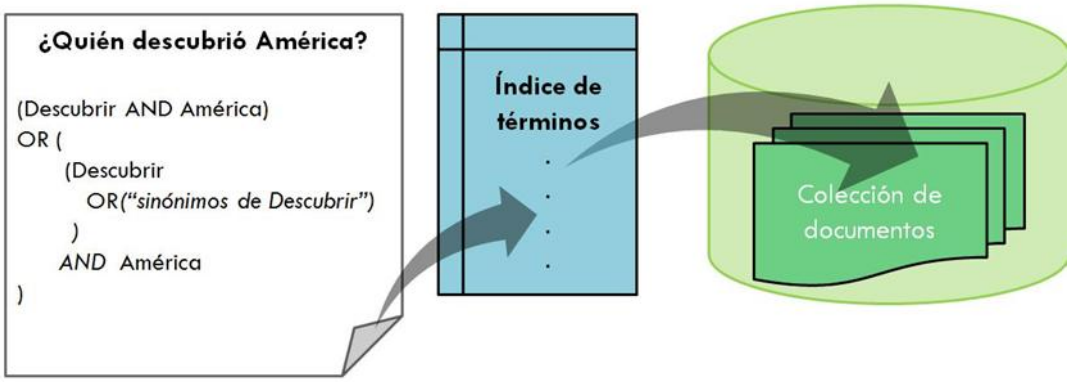


Ilustración 8: Búsqueda típica con índices de términos

La búsqueda de información, por tanto, de la aplicación, consiste en realizar consultas en un lenguaje capaz de recuperar datos coincidentes de la base de datos, con el formato de triplas (objeto – atributo – valor, formados mediante los elementos útiles que han sido extraídos en el apartado anterior. Una vez finalizada esta búsqueda, el sistema devolverá una lista de coincidencias, que a veces puede contener datos que no eran los que el usuario quería pero que no son incorrectos.

Por ejemplo:

¿Cuál es la población de Toledo?

Devolvería la población de Toledo (España), Toledo (Ohio) y Toledo (Colombia), y entraríamos en problemas de contexto que la aplicación es incapaz de resolver, con lo que muestra todos los valores coincidentes, y es el usuario quien se encarga de seleccionar de la lista el dato que buscaba.

De cualquier modo, el sistema tan solo ha de mostrar esta lista de coincidencias que contienen la o las respuestas coincidentes con los elementos extraídos previamente.

2.6 RDF

Los sistemas de QA utilizan un sistema de almacenamiento donde guardar la información. Se pueden distinguir a su vez dos tipos de repositorios de información: los estructurados, donde la información guarda una estructura fácilmente reconocible; y los libres donde se tiene la información almacenada sin estructura implícita. Esta última es cuando se tiene un gran volumen de información documental y no se conoce una forma de ordenarla, debido a que suelen ser textos o frases, para ello se utiliza un índice que facilite el acceso al párrafo correspondiente.

La mayor diferencia entre ellas es la distinta manera de realizar las

búsquedas que ofrecen cada una; en el caso de las libres, la exploración se realiza sobre el contenido, mientras que en las estructuradas al poseer diferentes campos delimitados por marcadores en su estructura se pueden realizar por campo o contenido del campo.

Para este proyecto se ha decidido utilizar las bases de datos estructuradas ya que la posibilidad de buscar por campo o por valor del campo es una de las necesidades de la aplicación. Estas bases de datos consisten en tener un esquema base sobre el que se apoya cada una de las instancias que se quieren almacenar, de ese modo se puede acceder de una manera rápida y sencilla a la información requerida; el punto en contra de este modelo, viene dado porque normalmente se tiene menos información que en las otras, o al menos en dominios más restringidos. Suelen almacenarse en bases de datos para utilizar el sistema de consultas de éstas.

El Resource Description Framework (RDF [20]) es un modelo de metadatos de XML, o una aplicación de metadatos que utiliza XML. Permite describir datos en sitios web, aunque su funcionalidad ha sido generalizada para todo tipo de datos, proveyendo las relaciones necesarias entre las aplicaciones para que puedan intercambiar información esquematizada gracias a este modelo en el lenguaje máquina.

El RDF y la Web Semántica están estrechamente relacionados, ya que es el tipo de dato recomendado por la W3C a utilizar en Web Semántica, para aportar información descriptiva sobre los datos que se utilizan.

La Web Semántica es una Web ampliada donde gracias a que contiene una información mejor definida, los usuarios tendrán más al alcance la respuesta a las preguntas que están buscando.

Se la llama Web Semántica por la utilización de una semántica que hace de infraestructura común, mediante la cual, es posible compartir y procesar información de forma sencilla. Esta inclusión de semántica, evita los problemas que pueden tener otras Web que carecen de ella, donde el acceso a la información puede llegar a ser más complicado.

El RDF es una buena manera de describir objetos con propiedades y las relaciones que tienen entre sí, ya que mediante un solo RDF se puede describir una relación entre dos objetos [21].

Hay tres tipos de representaciones asociadas a un RDF:

- Mediante un grafo, en el que se pueden ver los objetos y las relaciones entre sí a través de las flechas que los unen.

- Mediante el propio RDF, a través de triplas que unen objeto-atributo-valor.

- Mediante una representación XML equivalente, consiste en crear etiquetas de apertura y cierre con el atributo correspondiente y que encierren al valor.

2.6.1 Origen del RDF

La necesidad de un sistema como RDF [22] surge en agosto de 1997 dentro del Consorcio Web W3C, cuya actividad está relacionada con los metadatos y apoyada por entidades y empresas muy influyentes en el ámbito industrial; tales como creadores de navegadores específicos como Netscape o Microsoft, o motores de búsquedas. Se nutre tanto de iniciativas como de otros trabajos del W3C, y sobre todo de los trabajos en torno al Dublin Core (DC), que fueron los primeros modelos de metainformación que adoptaron y utilizaron la sintaxis del RDF.

Desde febrero de 1999, el consorcio Web emitió una recomendación, tras muchos intentos de acuerdo y borradores previos sobre la especificación del modelo y la sintaxis de RDF, y desde marzo del 2008, su esquema es una firme propuesta de recomendación.

2.6.2 Estructura de RDF

La estructura se fundamenta en tres componentes:

- Objeto: Un recurso, que puede ser una página web completa, o parte de una página Web, por ejemplo, un elemento HTML o XML específico dentro de la fuente del documento. Un recurso también puede ser un objeto que no es directamente accesible a través de la Web, por ejemplo, un libro impreso. Los recursos son siempre nombrados por URIs (Universal Resource Identifier, identifica a un objeto único dentro de un grupo) más identificadores opcionales. Cualquier recurso puede tener una URI; puede existir una URI para cualquier entidad imaginable.

- Atributo o propiedad: Un atributo o propiedad es un aspecto específico, característica, atributo, o relación utilizado para describir un recurso.

- Valor: Es el valor de un objeto para una propiedad determinada.

Estas tres partes se denominan, respectivamente, el *objeto,* el *atributo,* y el *valor.* El valor de una declaración, (o el valor de la propiedad) puede ser otro recurso o un literal, es decir, un recurso (especificado por una URI) o una cadena simple u otro tipo de datos simples definidos por XML.

Para comprender mejor la estructura se puede observar en este ejemplo extraído de Wikipedia.org [23]:

<http://en.wikipedia.org/Tony_Benn> <http://purl.org/dc/elements/1.1/title> "Tony Benn".
<http://en.wikipedia.org/Tony_Benn> <http://purl.org/dc/elements/1.1/publisher> "Wikipedia".

En ella se puede apreciar, el objeto *http://en.wikipedia.org/Tony_Benn*, que en este caso se trata de una página web; y que tiene dos atributos: *publisher* y *title,* que a su vez vienen determinados por una dirección web. Esta dirección web lo único que nos indica, es de dónde se toman esos atributos. Y por último, el valor que en este caso se trata de dos literales (por ir encerrados entre comillas).

Alternativamente, expresado en RDF/XML queda de la siguiente manera:

```
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/">
<rdf:Description rdf:about="http://en.wikipedia.org/Tony_Benn">
 <dc:title>Tony Benn</dc:title>
 <dc:publisher>Wikipedia</dc:publisher>
</rdf:Description>
</rdf:RDF>
```

En este ejemplo el RDF se ha adaptado al sistema de etiquetas de XML, donde se crea una etiqueta general *RDF* que contiene una etiqueta *Description*, que describe al objeto *http://en.wikipedia.org/Tony_Benn* mediante dos atributos *publisher y title,* que toman su dirección del prefijo dc especificado previamente, y entre sus etiquetas encierran el valor del atributo correspondiente.

Además de estas dos representaciones alternativas, los RDFs pueden ser modelados mediante un grafo que representa esta información de una manera visual más atractiva y que permite ver de manera más clara las relaciones entre objetos. Para el ejemplo anterior el grafo resultante sería:

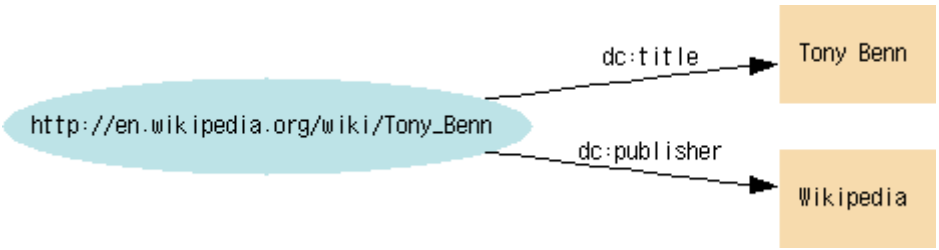


Ilustración 9: Ejemplo de grafo RDF

2.6.3 Búsquedas sobre RDF

De modo similar al lenguaje SQL en las bases de datos, se desarrollaron lenguajes de consulta para RDF, que permiten ejecutar búsquedas complejas sobre grafos RDF mediante una sintaxis sencilla.

A falta de acuerdo sobre un lenguaje estándar más o menos comúnmente aceptado, han surgido diferentes iniciativas:

- RDQL, por Hewlett Packard.

  RDQL [24] es un lenguaje de consulta para RDF basado en los modelos Jena. Es una implementación de SquishQL, que a su vez proviene de rdfDB. RDQL provee un sistema de creación de expresiones que son equiparadas a unas triplas de datos, y devuelve una lista de aciertos. No hay posibilidad de utilizar la disyunción.

- RQL, por el instituto ICS-FORTH de Grecia.

  RQL [25] es un lenguaje basado en XML. Las consultas se envían al servidor y son contestadas mediante una respuesta RQL (parecido a las solicitudes HTTP). Los nombres de etiqueta son siempre escritos en mayúsculas, mientras los nombres de atributo se escriben en minúsculas.

- SeRQL, por la empresa holandesa Aidministrator.

  SeRQL [26] (Sesame RDF Query Language, pronunciado como "circle") es un lenguaje de recuperación para RDF/RDFS desarrollado por Aduna como integrante de Sesame. Es una combinación de características de otros lenguajes (mayormente RQL, RDQL, N-Triples y N3) añadiendo otras propias. Está asociado con el programa Sesame, que es un framework para RDF de código libre y que soporta consultas. Tiene una sintaxis muy similar a la de RQL, añadiendo modificaciones para facilitar el análisis sintáctico

- SPARQL

  SPARQL [27] es la evolución del RDQL. Además es el lenguaje para Web Semántica recomendado por la W3C. Dado que es el lenguaje que se utiliza en el proyecto, se explicará con más detalle en el siguiente apartado.

2.7 SPARQL

2.7.1 Introducción

SPARQL (Simple Protocol and RDF Query Language), pronunciado “Sparkle”, es un lenguaje de recuperación para RDFs. Se trata de una recomendación para crear un lenguaje de consulta dentro de la Web semántica que está ya implementada en muchos lenguajes de programación

y bases de datos. Desde 2005 está en proceso de estandarización por el W3C.

Este lenguaje de consulta permite centrarse en la información que se desea tener recopilada, sin tener en cuenta la tecnología de la base de datos o el formato utilizado para almacenar esos datos. Debido a que las consultas en el lenguaje SPARQL expresan objetivos de alto nivel, es fácil extenderlos a orígenes de datos inesperados, o incluso transferirlos a nuevas aplicaciones.

Además, SPARQL ha sido diseñado para un uso a escala de la Web, así permite hacer consultas sobre orígenes de datos distribuidos, independientemente del formato. A la hora de recuperar información la creación de una sola consulta a través de diferentes almacenes es mejor que múltiples consultas.

SPARQL consiste en tres especificaciones complementarias, que contienen diferentes partes de su funcionalidad. SPARQL lo forman un lenguaje de consultas, un formato para las respuestas, y un medio para el transporte de consultas y respuestas:

- SPARQL Query Language: Núcleo de SPARQL. Explica la sintaxis para la composición de sentencias y su concordancia [28].

- SPARQL Protocol: Formato utilizado para la devolución de los resultados de las búsquedas (instrucciones SELECT o ASK), a partir de un esquema de XML [29].

- SPARQL Query XML Results Format: Describe el acceso remoto de datos y la transmisión de consultas de los clientes a los procesadores. Utiliza WSDL para definir protocolos remotos para la consulta de bases de datos basadas en RDF [30].

2.7.2 Componentes

El lenguaje SPARQL posee tres componentes importantes: URIs, literales y variables, procedentes del lenguaje RDF.

- URIs: sirven para especificar los recursos. Su especificación está en la RFC 3987 [31].

- Literales: se describen como una cadena de caracteres encerradas entre " ", y que representan objetos constantes elementales.

- Variables: estas variables son globales, además deben de ser prefijadas por "?" o "$" seguidas de un nombre cualquiera en el que no puede contener estos caracteres. Se utilizan para asignar valores de manera

temporal, para almacenar resultados o para utilizar filtros sobre ellas.

2.7.3 Sintaxis

La sintaxis de SPARQL es similar a la de RQL, añadiendo algunas modificaciones para facilitar el análisis sintáctico del lenguaje.

Básicamente se podría decir que son consultas en SQL, ya que siguen una estructura semejante:

PREFIX prefijo1, prefijo2..., prefijoX
SELECT/CONSTRUCT var1, var2,...varX
WHERE {condiciones}

El primer bloque (PREFIX) incluye los prefijos que serán utilizados en la consulta, de modo que se asigna a una combinación de letras una dirección. El segundo bloque (SELECT/CONSTRUCT) indica las variables de las que se desea conservar su valor tras ejecutar la consulta, o mejor dicho los valores que se desea retornar. El tercer bloque (WHERE) es donde está la consulta en sí, incluye las condiciones que se tienen que dar en los RDF para que seleccione esa tripla de valores como una coincidencia, y se almacenen las variables indicadas en el bloque anterior.

El lenguaje SPARQL tiene cuatro tipos de operaciones sobre los RDFs, a continuación se pueden apreciar las cuatro con un ejemplo explicativo.

- SELECT es la instrucción más común. Devuelve las variables que se le indican para los casos en los que coinciden con el patrón de búsqueda.

```
PREFIX foaf: <http://lenguajerecuperacionsparql.com/foaf/0.1/>
SELECT ?nameX ?nameY ?nickY
WHERE
{ ?x foaf:knows ?y ;
foaf:name ?nameX .
?y foaf:name ?nameY .
OPTIONAL { ?y foaf:nick ?nickY }
}
```

Ilustración 10: Sintaxis del operador SELECT

En este ejemplo, se observa que se declara un prefijo llamado *"foaf"* al que se le asigna una dirección web que hará de prefijo posteriormente. Mediante el SELECT se indica que se quiere saber el valor de las variables *?nameX*, *?nameY* y *?nickY.*

Dentro del bloque WHERE (encerrado mediante llaves), se observa las

tres condiciones que se le imponen a los RDFs resultantes, las dos primeras indican que se busca un objeto cualquiera que tenga los atributos *foaf:name* y *foaf:knows* (aquí se puede apreciar cómo se utilizan los prefijos declarados previamente), y de los objetos que hayan sido devueltos como valor del atributo *foaf:knows*, se buscan aquellos que tengan nombre.

En esta consulta aparece el bloque OPTIONAL dentro del WHERE, en este caso la función de OPTIONAL, es evitar que lo que se especifique dentro sea una condición, convirtiéndolo en algo opcional; que si aparece es añadido en los resultados.

- CONSTRUCT devuelve un grafo RDF construido por la sustitución de variables en un conjunto de tres plantillas.

```
PREFIX foaf: <http://lenguajerecuperacionsparql/foaf/0.1/>
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
CONSTRUCT { <http://lenguajerecuperacionsparql.org/person#Alice> vcard:FN ?name }
WHERE { ?x foaf:name ?name }
```

Ilustración 11: Sintaxis del operador CONSTRUCT

- ASK devuelve un valor verdadero o falso indicando si se ha encontrado algún valor coincidente con los patrones de la consulta. Realmente es un SELECT simplificado ya que no devuelve los casos que han coincidido.

```
PREFIX foaf: <http://lenguajerecuperacionsparql/foaf/0.1/>
ASK { ?x foaf:name "Alice" }
```

Ilustración 12: Sintaxis del operador ASK

- DESCRIBE devuelve un grafo RDF representativo sobre las relaciones entre los objetos que enlazan con el especificado en la instrucción, describiendo estos recursos encontrados.

```
DESCRIBE <http://example.org/>
```

Ilustración 13: Sintaxis del operador DESCRIBE

2.7.4 Funcionamiento

Las consultas SPARQL cubren tres objetivos:

- Extraer información en forma de URIs y literales.
- Extraer sub-estructuras RDF.
- Construir nuevas estructuras RDF partiendo de resultados de consultas.

Un inconveniente para este lenguaje es la falta de posibles acciones sobre los datos, ya que únicamente pueden realizarse operaciones de lectura, lo que limita al sistema a leer y comparar datos. En el siguiente grafo se puede observar una estructura en formato RDF, sobre la que el sistema puede realizar consultas.

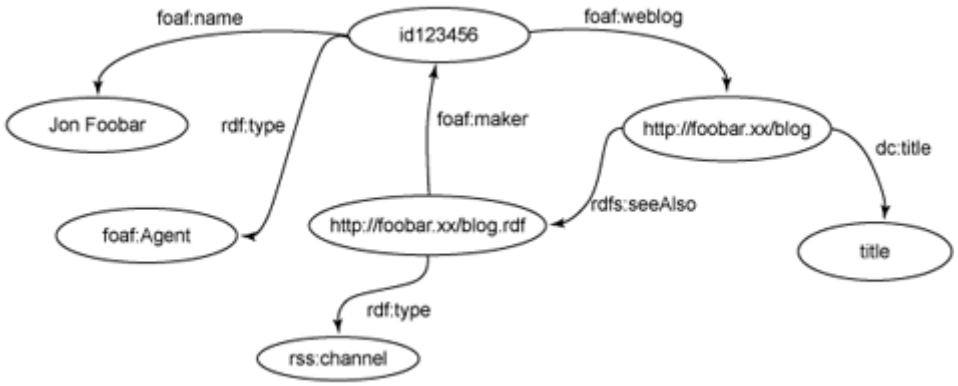


Ilustración 14: Modelado en forma de grafo de una estructura RDF

Una consulta simple sobre este grafo sería la mostrada en la figura siguiente [32] :

```
PREFIX foaf:http://foobar.xx/
SELECT ?url
FROM
WHERE {
?contributor foaf:name "Jon Foobar".
?contributor foaf:weblog ?url .
}
```

Ilustración 15: Ejemplo de consulta SPARQL

En ella se desea encontrar la URL de una persona llamada "*Jon Foobar*". Como se puede ver la manera de referirse al objeto que cumple la condición (en este caso que su nombre sea igual a "*Jon Foobar*"), ha de ser el mismo que del que se va a extraer el dato requerido (en este caso la dirección del Weblog).

En la primera línea encontramos la palabra clave PREFIX. Su función es equivalente a la declaración de namespaces en XML: asociar una URI a una etiqueta, que se usará en adelante para describir el namespace. Pueden definirse tantas de estas etiquetas en una misma consulta como se necesiten o deseen.

El comienzo de la consulta queda marcado por la palabra clave SELECT. Semejante a su uso en SQL, sirve para definir los datos que deben ser devueltos en la respuesta. En el ejemplo tan sólo la URL. La palabra clave FROM identifica los datos sobre los que se ejecutará la consulta, una consulta puede incluir varios FROM.

La palabra WHERE indica las restricciones de los RDF, que para este caso se obliga a que el objeto *?contributor* tenga un atributo *foaf:name,* equivalente al literal *"Jon Foobar",* y se solicita el atributo *foaf:weblog* de los casos que coinciden.

3. Diseño del sistema

3.1 Introducción

El proyecto consiste en la realización de un sistema de QA que emplea una base de datos con información estructurada de tipo semántica, donde pueden ser cargados datos de diferente índole. Esta información es utilizada como punto de partida para los objetivos de la aplicación.

El objetivo principal consiste en ser capaz de traducir el lenguaje natural utilizado por cualquier persona a uno que el sistema entienda y pueda interpretar para poder responder a una pregunta que le sea introducida por parte del usuario, sin tener que adaptar la formulación de la pregunta, debido a que va dirigida a una máquina. Esta correspondencia es el fundamento del sistema ya que es el traductor entre hombre – máquina, y una incorrecta transformación devolvería unos resultados incorrectos.

La aplicación puede responder a dos tipos de preguntas:

a) Preguntas simples: Consisten en preguntas que el sistema traduce, realiza la consulta y devuelve el dato que el usuario espera.

   Ej:
   ¿Quién es el presidente de Venezuela?

b) Preguntas complejas: Son preguntas simples encadenadas, se muestra al usuario los datos encadenados según el sistema los ha ido consultando.

   Ej:
   ¿Dónde nació la persona que gobierna Venezuela?

Todo sistema de QA necesita de una fuente de información, de la cual se generan las propuestas que serán mostradas al usuario. El sistema descrito es de propósito general y puede utilizar cualquier tipo de datos almacenado. Aunque para la evaluación del funcionamiento se han elegido unos datos para crear un escenario de prueba. Este escenario se explicará con detalle en la fase de pruebas, donde se analizarán los datos extraídos para probar el algoritmo y el correcto funcionamiento de la aplicación.

El siguiente gráfico muestra un esquema del funcionamiento de la aplicación y las relaciones entre los distintos bloques en los que está dividido.

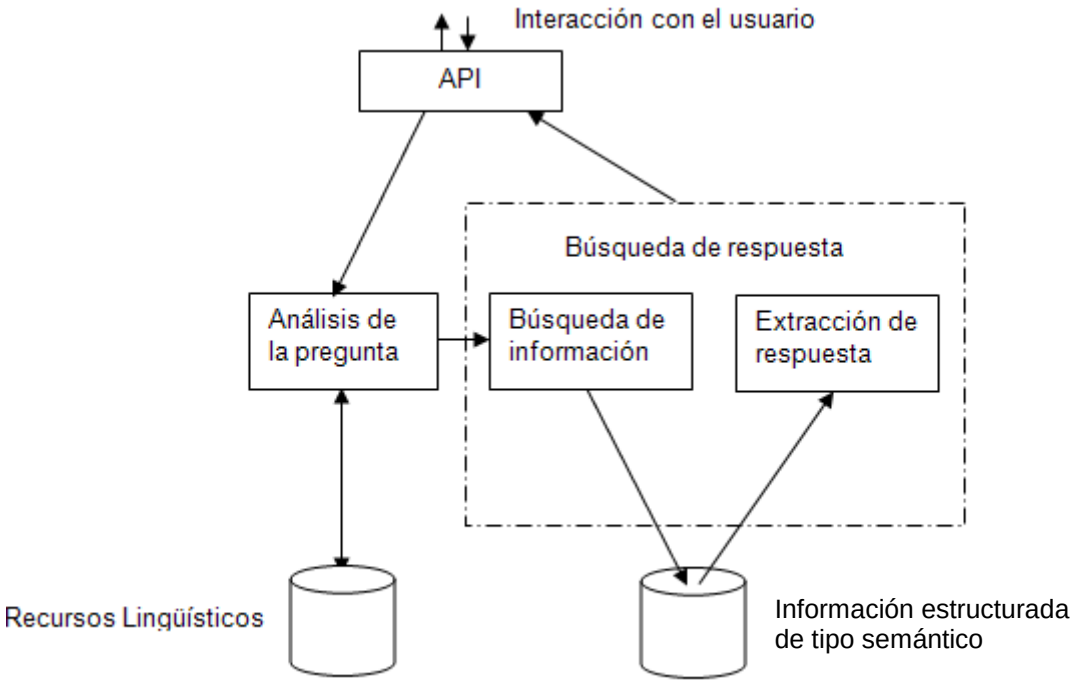


Ilustración 16: Esquema de un sistema de QA

3.2 Sistema de QA

El sistema consiste en una aplicación cuyo objetivo es conseguir dar una respuesta lo más correcta y concreta posible a las preguntas que le sean introducidas por el usuario. Las preguntas que soporta este sistema son preguntas en castellano sobre cualquiera de los contextos de la base de datos.

El sistema sigue la estructura más común de los QA, mencionada con anterioridad, con lo que se podría dividir en 3 grandes bloques como se aprecia en la ilustración anterior.

La aplicación parte de un análisis de la pregunta introducida, en el que se identifican las palabras de cada frase, eliminando de este modo las que no aportan información y "apilando" las demás para su posterior extracción.

Ejemplo:

- *¿Cuál es la población de Madrid?*

"Cuál", "población" y "Madrid" son las palabras que aportan información al sistema mientras "es", "la" y "de" son desechadas.

Una vez se han extraído las palabras, el sistema intenta identificar las entidades sobre las que se preguntan (mediante un reconocimiento de objetos), y la información que se desea conocer (asimilando cada posible palabra "útil" con las conocidas sobre los datos de las entidades), estableciendo una correspondencia entre la información, la base de datos y el lenguaje natural.

Ejemplo:
- *¿Cuál es la población de Madrid?*

Tras desechar las palabras sin información:
Cual – Pronombre interrogativo
Población – Atributo
Madrid – Candidato a objeto

Partiendo de esta clasificación el sistema generará la consulta pertinente.

Mediante este planteamiento, el sistema es capaz de responder a preguntas simples (cuando se solicita una característica de una instancia; ejemplo: la fecha de nacimiento de un actor) y complejas (cuando se pregunta por una propiedad del valor de un atributo de un objeto; ejemplo: la población del lugar de nacimiento de un político).

De este modo en las preguntas simples, el sistema se limita a "preguntar" por un atributo de una instancia identificada por el nombre de una determinada clase.

Mientras que en las complejas, enlaza y busca, gracias a las posibilidades que otorgan las consultas en SPARQL [33], donde es fácil solicitar las instancias que satisfagan determinado enlace entre dos de sus atributos.

Internamente la aplicación se basa, a grandes rasgos, en los ficheros de configuración para realizar la traducción entre lenguaje natural y el lenguaje de consulta:

- Fichero de términos: Alberga la expansión de sinónimos y lematización de cada una de las palabras que pueden estar relacionadas con los atributos de cada una de los objetos. Es decir, se trata de convertir a una palabra todos los sinónimos, formas verbales y variaciones de las palabras con las que se puede identificar cualquier propiedad de las conocidas para cada tipo de objeto.

  Ej:
  morir → *fallecido*|*fallecio*|*fallece*|*fallecimiento*|*fenecido*|*fenecio*|*fenece*

- Fichero de atributos: Contiene relaciones entre los atributos de los

objetos y los lemas del fichero de términos. Es decir, asigna a cada lema irreducible un atributo asociado de los objetos almacenados en la base de datos.

Ej:
birth_date → *nacer&fecha*

Posteriormente se ahondará más en estos ficheros pues recae sobre ellos una gran importancia de la aplicación.

Una vez se consigue encontrar lo que se buscaba, el sistema se encargará de procesar los datos y generar una salida para que el usuario pueda visualizar lo que solicitó.

En los posteriores apartados se explicará con mayor detenimiento cada fase de la aplicación, y qué se hace en cada uno de estos tres bloques en los que se divide.

3.2.1 Bloque 1: Análisis de la pregunta

El primer paso de cualquier sistema de QA, es conseguir "comprender" la pregunta y entender qué información está esperando el usuario que le sea devuelta.

Se parte de la premisa de que las preguntas introducidas están correctamente formuladas y bien escritas, inicialmente en idioma castellano[1].

Para la entrada de datos se han tomado algunas decisiones que intentan simplificar el desarrollo inicial de la aplicación, tales son:

- No distinción de tildes:
  Es el único error ortográfico que soporta el sistema ya que para evitar problemas de codificación, se eliminan las tildes de las palabras. Por ejemplo, sería lo mismo buscar *Leganes* o *Leganés*.

- Lematización de las palabras que son identificadas como posibles atributos:
  Ya que el sistema utiliza la palabra como unidad mínima posible, no distingue errores de tipo sintáctico, como pueda ser una mal concordancia de género o número entre las palabras o la omisión de elementos necesarios en la frase que jueguen el papel de nexo.

1 En principio el sistema podría ser multilingüe si se cambian los ficheros de configuración para la traducción de la pregunta, pero en el proceso se realiza cierto procesamiento lingüístico específico del español.

Ej:
¿Cuál es el población de Getafe?

- Reconocimiento de entidades multipalabra:
  Se basa en la premisa de partida de la correcta ortografía, incluyendo por lo tanto la primera letra en mayúsculas de los nombres propios.

  Cuando se detecta por tanto una palabra con la primera letra en mayúsculas, se le asigna la etiqueta de concepto. Si la siguiente palabra es otra palabra con la primera en mayúsculas, se realiza el mismo etiquetado y se concatena a la anterior, asumiendo así que se trata de un concepto formado por más de una palabra.

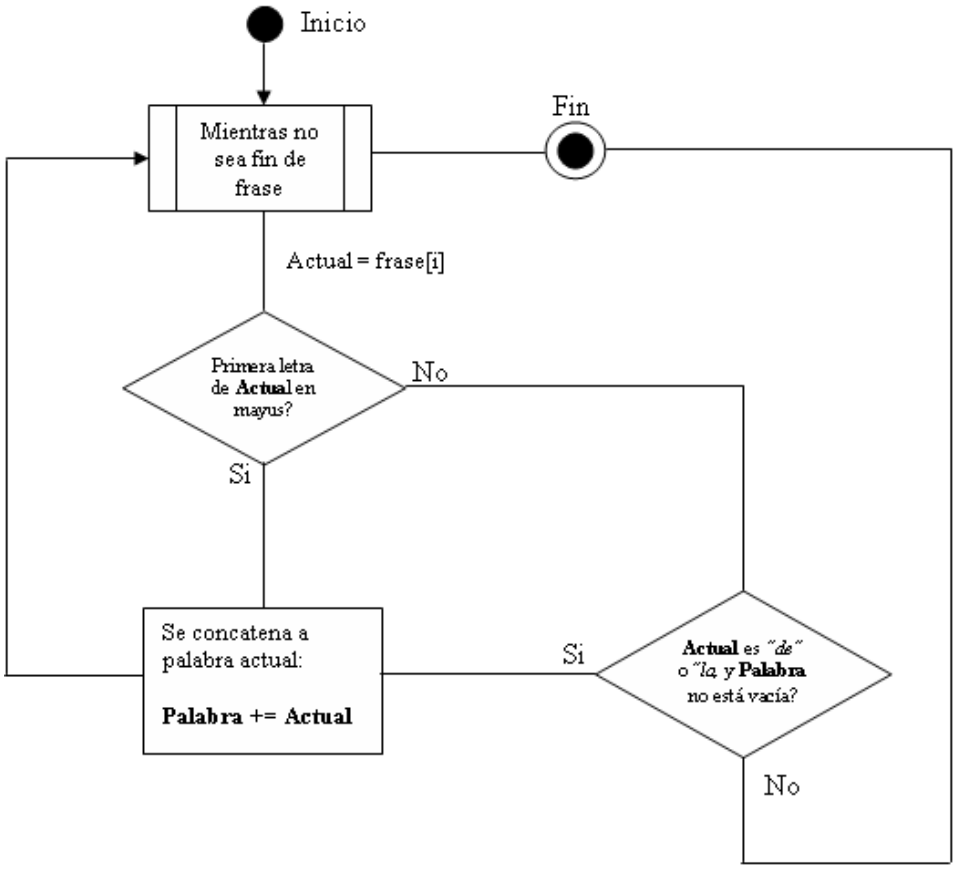


Ilustración 17: Flujo de identificación de multipalabras

La inclusión de las palabras "*de" y "la"* en las multipalabras se debe a intentar abarcar la máxima cantidad de nombres propios como por ejemplo: pueblos y ciudades españolas que en su nombre aparece "*de la*", ("*Villanueva de la Cañada*") o solamente "*de"* como es el caso de "*Alcalá de Henares*". Aunque no aparezca reflejado en el gráfico anterior, el orden de las palabras sí se tiene en cuenta, esto es: si se encuentra "*Villanueva de de la cañada*", se descartarían los "*de*" y se

buscaría tan solo *Villanueva.*

Ejemplo:

- *Villanueva de la Cañada*

Paso 1: *Actual = Villanueva*
Primera letra en Mayúsculas? → Sí
Palabra = Villanueva

Paso 2: *Actual = de*
Primera letra en Mayúsculas? → No
Palabra actual es "de","la" o "del"? → Sí
Palabra = Villanueva de

Paso 3: *Actual = la*
Primera letra en Mayúsculas? → No
Palabra actual es "de","la" o "del"? → Sí
Palabra = Villanueva de la

Paso 3: *Actual = Cañada*
Primera letra en Mayúsculas? → Si
Palabra = Villanueva de la Cañada

Paso 4: Fin

- Separación de subordinadas por pronombres interrogativos o relativos[2]:
  El sistema utiliza los pronombres, bien sea interrogativo o relativo, para marcar las preguntas, es por ello que cada vez que se lee un interrogativo/relativo, se genera una subpregunta anidada, permitiendo procesar así las mencionadas anteriormente preguntas complejas.

  Ej:
  ¿Quién es el alcalde del lugar en el que nació Rafa Nadal?

Una visión superficial del funcionamiento de este bloque puede contemplarse en el siguiente grafo.

2 Esto es un caso del anterior procesamiento lingüístico dependiente del idioma; en principio francés, italiano, portugués y otros derivados del latín componen las subordinadas de forma similar, e incluso en inglés, pero no es posible asegurarlo al 100%.

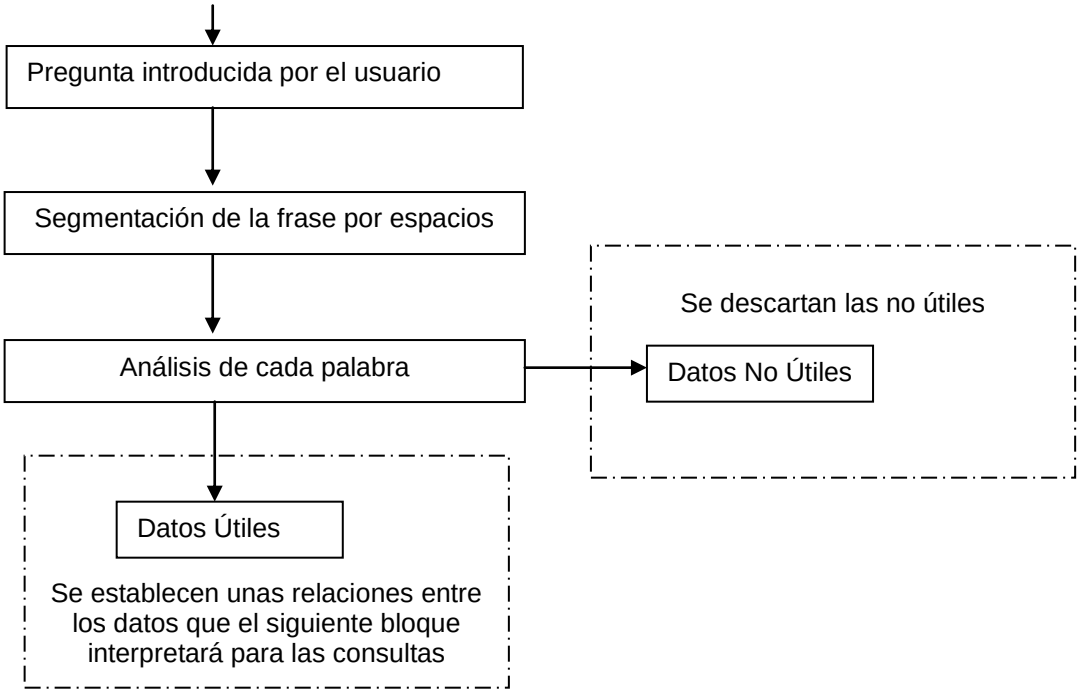


Ilustración 18: Algoritmo de análisis del etiquetado

Ejemplo:

- *¿Cuál es la población de la ciudad en que nació Rajoy?*

Lo primero que realiza el sistema es leer la pregunta que ha introducido el usuario, y convertirla del lenguaje natural a un formato conocido. Para ello, y dado que la frase puede contener más de una cuestión encadenada, se utiliza el siguiente método de fraccionamiento de la pregunta:

1. Se buscan los pronombres interrogativos/relativos en la frase que ha sido introducida. En caso de no encontrar ninguno la aplicación lo entiende como que no se requiere de su funcionalidad y confirma la afirmación escrita por el usuario.

Ejemplo:

- *¿Cuál es la población de la ciudad en que nació Stallone?*

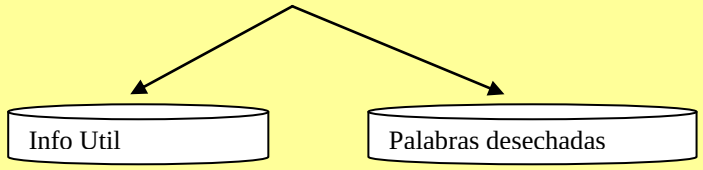
Pronombres relativos e interrogativos

2. Por cada pronombre del apartado anterior, se crea una nueva subcadena desde el pronombre hasta el siguiente pronombre o hasta fin de frase.

Ejemplo:
Consulta 1: Cuál → es la población de la ciudad
Consulta 2: en qué → nació Stallone

3. Cada palabra que no sea pronombre, se analiza y se comprueba si aporta algún tipo de información, desechándola en caso negativo, y almacenándola en una estructura de datos, en caso contrario.

Ejemplo:
Consulta 1: es la población de la ciudad
Consulta 2: nació Stallone

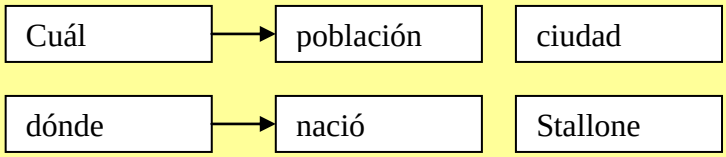


{ [población, ciudad], [nació, Stallone]}

{es, la, de, la}

De este modo, quedará una matriz de preguntas cortas que se le irán pasando al siguiente módulo de una en una, para ser interpretadas de manera sencilla y con la menor ambigüedad posible.

Así quedaría la estructura al finalizar este bloque:



La estructura de datos que se pasará al próximo bloque se va generando de acuerdo al siguiente esquema:

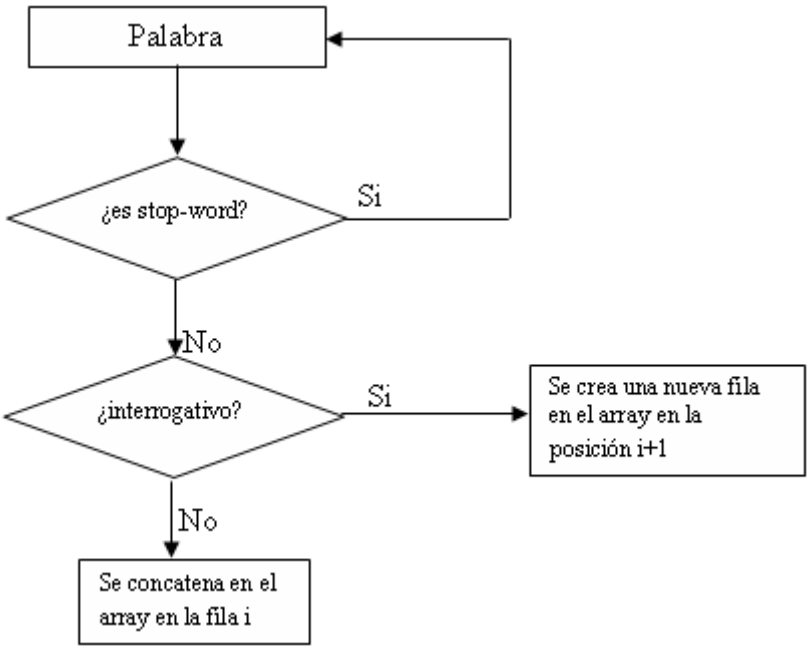


Ilustración 19: Algoritmo de división de la pregunta

3.2.2 Bloque 2: Búsqueda de información

El siguiente bloque se encarga de recoger la estructura de datos que contiene las palabras que poseen, o pueden poseer, información y las interpreta, generando y ejecutando consultas en la base de datos para intentar obtener esa información que está siendo solicitada. Un primer acercamiento al algoritmo de funcionamiento podría ser el mostrado en el gráfico siguiente.

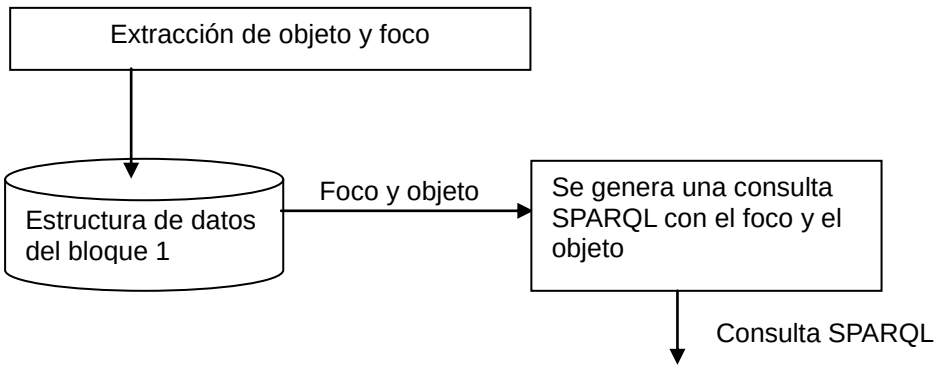
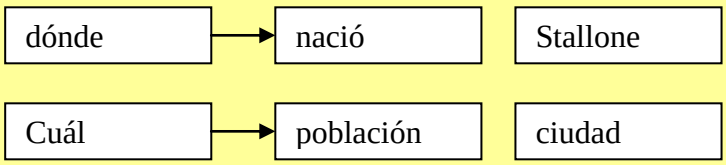


Ilustración 20: Algoritmo de la búsqueda de información

Partiendo del bloque anterior, que ya se han determinado el número de preguntas en la oración, y están fraccionadas del modo anteriormente

mencionado, se toma la última pregunta de la frase, que será la primera que deba resolver el sistema, ya que las anteriores estarán relacionadas, y por lo tanto se necesita ir recabando información desde el final, para ir contestando las preguntas previas.

(Siguiendo el ejemplo anterior)
Al permutar las posiciones se responderían en este orden



Esta información se obtendrá a partir de los pronombres interrogativos/relativos y palabras estrechamente relacionadas con los atributos para realizar la traducción de palabras y conceptos de la base de datos.

Para lograr encontrar estos datos, la extracción de palabras claves está basada en los ficheros externos mencionados con anterioridad, el fichero de términos y el de atributos (ficheros sobre los que recae un gran porcentaje de funcionalidad del sistema).

Los ficheros tienen unas estructuras definidas para que el sistema pueda analizar los datos que contienen de una manera sencilla y eficaz a la hora de encontrar el foco de la pregunta.

Estos dos ficheros se utilizan para: almacenar la lematización de las palabras que intervienen en los datos y expansión de sinónimos, mientras que el otro que se encarga de comprobar, con los datos que se tiene, qué puede estar solicitando el usuario.

- El fichero de términos está constituido con el siguiente formato:

  Forma \t *Lema1* | *Lema2* | *Lema3* |…

  Donde las distintas opciones para ese lema vienen separados por barras verticales, indicando que cualquiera de las posibilidades indicadas se corresponde con la palabra clave de la parte de la izquierda.

  Ej:
  nacer \t *nacido* | *nace* | *concebido* | *nacio*

- El otro fichero, el de atributos está formado por un número de entradas

equivalentes al número de atributos distintos que se manejan en la aplicación:

Atributo \t Expresión1 | Expresión2| Expresión3 | …

Las posibilidades de la parte derecha del tabulado, contiene las distintas maneras de las que se puede estar solicitando cierto atributo, separadas entre sí por barras verticales de nuevo, que en las expresiones regulares significa disyunción. Cada una de estas posibilidades puede estar regida por varias condiciones a su vez.

Ej:
birth_date \t nacer&fecha

Partiendo de estos ficheros, el sistema ha de buscar dos términos de los tres posibles en las estructuras RDF de los conceptos de la BD: objeto, atributo, o valor, para así devolver al usuario el que le falte. Estos elementos pueden ser de los siguientes tipos:

- Lema normalizado: Es el tipo dato que busca el usuario sobre un determinado objeto.

- Candidato a objeto: Objeto que contiene unos atributos relacionados que pueden ser obtenidos mediante los lemas normalizados. Son candidatos ya que no se pueden garantizar que sea un objeto o valor.

- Valor de atributo: Información que se tiene en la base de datos sobre un objeto y una determina propiedad.

Ejemplo objeto-atributo-valor (OAV):

Una posible relación sería:
Madrid →alcalde → Gallardón

Si en la pregunta, aparece alcalde y Gallardón, se sabe que el dato buscado es Madrid. Obviamente el atributo ha de aparecer acompañado o de un objeto o de un valor, puesto que sino, no se sabría sobre qué se está preguntando.

El proceso a seguir para cada consulta (o subconsulta en caso de ser encadenadas) consta de los siguientes pasos:

1. Se busca el foco de la pregunta: Esta búsqueda se basará en el tipo de pregunta que se le haya realizado, y lo primero que se hace es extraer el pronombre para ver qué tipo de información se espera en el retorno.

De modo que si se pregunto "*quién*" se le está requiriendo información de una persona. Esta información sirve para desambiguar sobre el atributo solicitado, pero solo con esto no se puede conseguir lo esperado, por lo tanto se deberán buscar otras "pistas" que ayuden a identificar.

Ejemplo:

- *¿Cuándo fue la fecha de nacimiento de Rajoy?*

Cuando → Se está pidiendo una fecha

2. Una vez se sabe el tipo de dato que se espera que devuelva el sistema se procede al etiquetado de las palabras que han sido catalogadas como poseedoras de información.

   La primera tarea es dividir la consulta en tokens (unidad mínima de información que puede ser analizada por sí sola), y comprobar qué tipo de palabra es. Para ello se realizan las siguientes comprobaciones:

   I. Se busca en el fichero de términos si la palabra es alguno de los lemas correspondiente a alguna forma de las conocidas. Esta comparación se realiza mediante expresiones regulares que de una manera rápida y sencilla devolvería la lematización atribuida a dicho lema, en caso de que exista en el fichero.

      En caso de que la palabra no devuelva ningún lema, esta palabra es descartada como palabra que aporta información y se procede al siguiente paso. En el caso contrario, para cada consulta se crea una estructura de datos que va recogiendo los lemas que devuelven las palabras con información para que una vez finalizado el proceso de análisis y etiquetado, se proceda a la búsqueda de respuesta.

Ejemplo:

- *¿Cuándo fue la fecha de nacimiento de Rajoy?*

Fue -> lema:ser
la -> descartada como palabra sin información
fecha ->lema:fecha
de -> descartada como palabra sin información
Rajoy -> descartada como palabra sin información

(Aunque Rajoy en la siguiente pasada se seleccionaría como palabra con información)

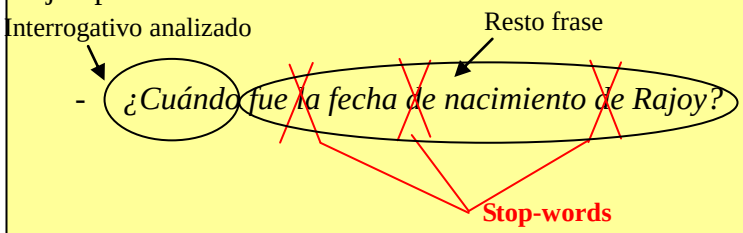
II. Las palabras que no devuelven ningún valor, es decir, que no encajan en ninguna de estas comparaciones, se les realiza otra prueba alternativa antes de descartarlas, ya que aunque no aporte información al sistema de esta manera, son candidatas a ser identificadas y etiquetadas como posibles objetos o valores.

    Esta nueva comprobación simplemente consiste en comprobar si la palabra empieza por letra mayúscula, de este modo se puede saber si se trata de un nombre propio, lo que implicaría que se trataría de un candidato a objeto o a valor de atributo (esta distinción será tratada con mayor profundidad en el apartado de implementación).

III. Si la palabra no ha sido etiquetada en los anteriores casos, o bien como lema de una forma normalizada, o bien como candidata a objeto, se realiza una última prueba. Si se trata de una palabra formada únicamente por dígitos, es decir un número, se le etiqueta únicamente como valor de objeto, ya que un objeto que fuera un número, véase una fecha o un siglo, carecería de atributos o tendría muy pocos, con lo que siguiendo la definición de objeto, no sería muy producente crear un objeto vacío.

Las palabras que no estén etiquetadas una vez realizadas estas tres comprobaciones son descartadas de la oración, ya que no se utilizarán en las fases siguientes.

Ejemplo:



fue → devolvería su lema: ser

fecha → devolvería su lema: fecha

nacimiento → devolvería su lema: nacer

Rajoy → al no devolver ningún lema y estar la primera en mayúsculas, será candidata a objeto.

Con lo que tras el análisis:

[objeto] = Rajoy
[palabras_clave] = {fecha, nacer}

3. Se seleccionan atributos a partir de las palabras clave: Partiendo de la lista de palabras claves, se accede al fichero de atributos para seleccionar los que son solicitados por el usuario. Se vuelve a realizar el mismo proceso que en el paso anterior, es decir, a través de expresiones regulares se va comprobando si existe algún atributo que contenga alguna expresión n-ésima que coincida con las palabras clave sobre las que se va iterando.

   Si se cumple, se retorna la propiedad, para posteriormente ser añadida a la búsqueda.

   Gracias a estas relaciones la desambiguación lograda es bastante potente, aunque siempre suponiendo que se parte de preguntas correctamente formadas.

Ejemplo: (Siguiendo el anterior)

- *¿Cuándo fue la fecha de nacimiento de Rajoy?*

[objeto] = Rajoy
[palabras_clave] = {fecha, nacer}

Extracto del fichero de atributos:

Author \t pintar|escribir|crear

Birth_place \t nacer&lugar

Birth_date \t nacer&fecha

No coincide con las palabras clave → Avanza al siguiente atributo

Se almacena en los atributos a buscar

Coincide con lo que tiene palabras_clave

[Atributos] = {Birth_date}

4. Una vez se tiene identificados los atributos de la consulta, falta el foco sobre el que buscar la información, para ello se buscan palabras que empiecen por mayúsculas o sean números seleccionadas como candidatos previamente, ya que si se trata de alguna perteneciente a estas categorías, tiene que ser un nombre propio, por lo que tendrá que estar capitalizada o estar formada por dígitos.

   Esto lleva a un problema ya que en el caso de que alguien pregunte "¿Quién nació en Roma?" no es lo mismo que "¿Cual es la población de Roma?", y en ambos casos se trata de la entidad sobre la que se requiere información, pero no del objeto.

   Para evitarlo, se utiliza la información del atributo y se comprueba si ese atributo es propio de la entidad (en este caso de un lugar), en el caso de que no sea de ese tipo de entidad, se buscaran los atributos cuyo valor sea esta entidad. Por otra parte, si pertenece, la consulta generada será la básica y buscará en la entidad proporcionada, dicha propiedad.

Ejemplo 1: Entidad = Objeto

- *¿Cuál es la población de Roma?*

[objeto] = Roma
[palabras_clave] = {población}

Comprobando en el fichero de términos:
Population \t población

Extrayendo del fichero de atributos:
Population \t population

[Atributos] = {Population}

Se busca una entidad llamada "*Roma*" que tenga un atributo denominado "*population*". Como en este caso lo encuentra, devuelve el valor:

2.726.539 habitantes

Ejemplo 2: Entidad = Valor de un objeto desconocido

- *¿Quién nació en Roma?*

[objeto] = Roma
[palabras_clave] = {nacio}

Comprobando en el fichero de términos:
nacer \t nacio

Extrayendo del fichero de atributos:
birth_place&birth_date \t nacer

[Atributos] = {Birth_place|Birth_date}
**Nota: Como no se genera suficiente información, el programa devuelve las dos propiedades en un principio.*

Se buscan objetos que tengan una propiedad llamada "*Birth_place*" o "*Birth_date*", y cuyo valor sea "*Roma*"

Como se puede ver en el ejemplo 2, que una entidad sea etiquetada

como objeto no quiere decir que sea el objeto, ya que puede ser valor de un atributo. Por lo tanto, hasta que no se realice la consulta no se puede comprobar de qué tipo es la consulta, por ello el sistema lanza primero la que se ve reflejada en el ejemplo 1, y en caso de devolver un valor vacío generará una nueva que compruebe si existen objetos que en sus atributos contengan el posible objeto.

Hay una tercera posibilidad y es que el atributo se encuentre en ambas posiciones, en este caso se generan dos consultas paralelas y el programa decidirá por el peso de la pregunta cuál será la opción de vuelta, aunque sin descartar ninguna y mostrando al usuario todas las que se hayan obtenido.

3.2.3 Bloque 3: Extracción de la respuesta

Este último bloque se encarga de gestionar las consultas que se formulan a la base de datos.

Partiendo de la estructura de datos resultante del apartado anterior, este bloque se encarga de traducirla al lenguaje SPARQL, lanzar la consulta a la base de datos por la que será interpretada y recoger el resultado que se generará para posteriormente mostrarlo al usuario.

La traducción se basa en el concepto de objeto-atributo-valor (OAV). Partiendo de él, se equiparán los OAV contenidos en la fuente de información con otro OAV con incógnitas en el campo de valor o de objeto, ya que no se puede asegurar en ningún caso, de cuál de los dos se trata.

Ej:
Este ejemplo, es el correspondiente a la tupla OAV que resultaría de la pregunta *"¿Cuál es la población de Madrid?"*

(?x-- población -- Madrid ||
Madrid -- población -- ?x)

(Siguiendo la sintaxis Objeto -- atributo -- valor)

?x → es la incógnita que el sistema deberá devolver
Población → sería el atributo que ha sido extraído en el bloque anterior para buscar en la consulta
Madrid → puede ser el valor del atributo o el objeto. De ahí la disyunción que siempre es contemplada.

En este apartado, se intenta dar una idea del concepto utilizado a la hora de realizar las consultas, en el apartado de implementación se profundizará en cómo quedarían las consultas en el lenguaje SPARQL y cómo es tratada por la base de datos.

En el siguiente diagrama se puede apreciar la secuencia de ejecución que sigue.

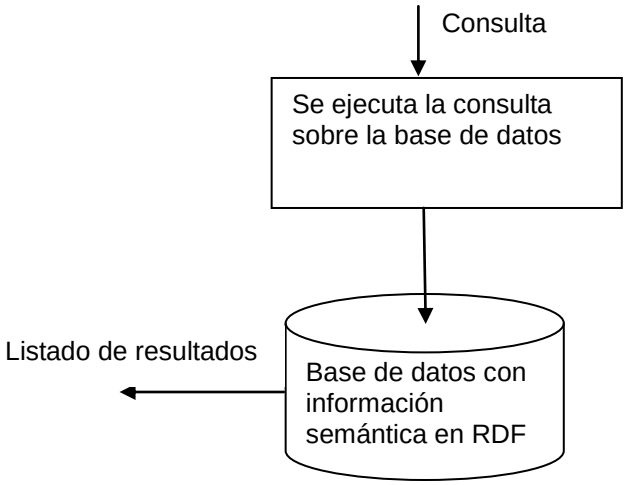


Ilustración 21: Ejecución de la extracción de la respuesta

Una vez que ya está completamente formulada la consulta en SPARQL, el sistema se conecta a la base de datos, en la que previamente se han cargado los ficheros que contienen la información de los objetos, y se realiza al sistema una petición de consulta.

Esta consulta realmente consiste en la unión de otras dos. Una de ellas será, de acuerdo a lo explicado en el paso anterior, la que se encargue de buscar el objeto por el nombre, pero hay casos en los que no se identifica del modo que el usuario especifica (un mote, por ser más conocido por el apellido..), así que para evitar que se devuelvan fallos o preguntas no contestadas cuando realmente se tiene la información, se añade a la consulta, si bien no siempre es necesario, otra comprobación referente al nombre del objeto, para asegurarse que no se encuentra entre los variantes del nombre también almacenadas (alias, apellido, mote…)

Ejemplo:

-Donde nació Olvido Gara?

Aunque realmente sea su nombre, al ser más conocida como Alaska. Wikipedia asigna Alaska al atributo identificativos de nombre. Pero si alguien se refiere a la cantante por su nombre de nacimiento; el programa debería identificarlo y devolver el valor correcto como si se hubiera preguntado por su identificador **"***Alaska***"**

México D.F
Seria el valor devuelto en ambos casos.

La consulta generada contempla además otro caso al básico, por un lado se comprueba si existe un objeto que tenga por nombre el foco, y por otro lado se busca un atributo cuyo valor sea el foco. Dado que esta distinción es muy compleja, es más fácil filtrar los datos de este modo que generar reglas o funciones que intenten discernir en qué caso representa cada cual de los casos.

El sistema se comunica con la base de datos y recibe un array con una estructura determinada por ARC2, sobre el cual se puede extraer tanto los datos, como el tiempo de consulta o la estructura sobre la qué han sido extraídos.

Una vez terminada la consulta, el sistema procesa los datos y dependiendo del tipo de pregunta que se haya realizado, muestra el resultado final de la consulta o lo añade a la siguiente pregunta que se va a realizar. El siguiente diagrama muestra el comportamiento.

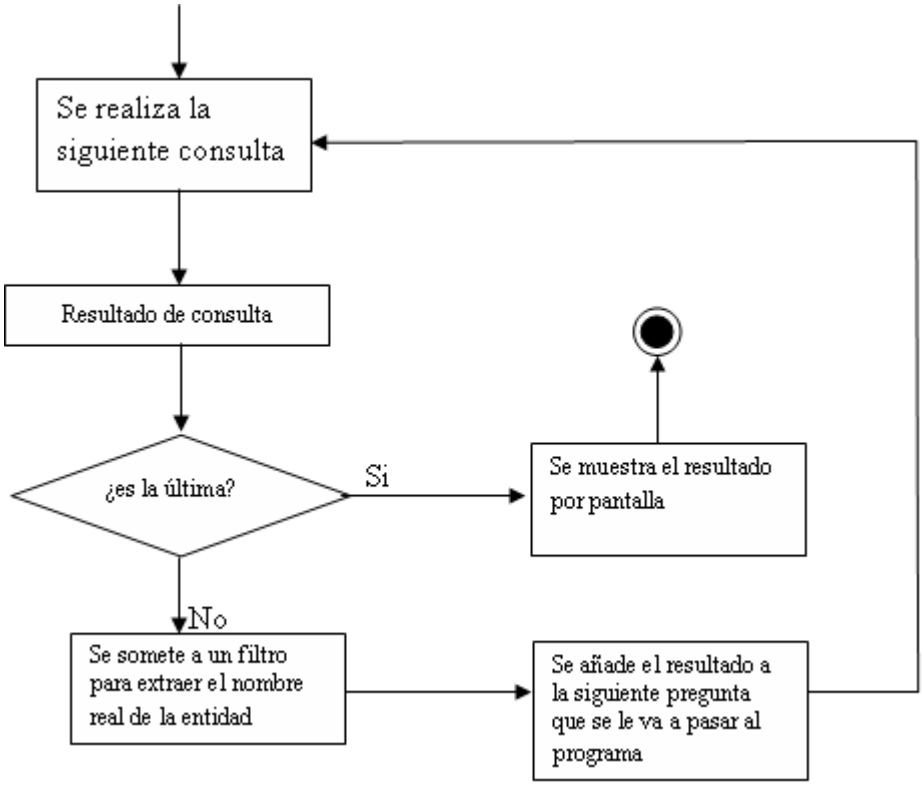


Ilustración 22: Algoritmo de funcionamiento de la aplicación

Ej: Un usuario solicita "¿Cuál es el alcalde de la capital de Francia?". El sistema solo mostrará "Bertrand Delanoë" sin previa comunicación de que la capital de Francia es París.

Esta concatenación es sencilla ya que solamente consiste en añadir el resultado obtenido al candidato a objeto de la siguiente pregunta, de este

modo en la siguiente pregunta solamente se buscará el atributo, dado que ya se dispone del objeto

3.2.4 Bloque 4: Interfaz

Para la interacción con el usuario, se ha creado una interfaz a modo demostrador donde se le permita formular las preguntas al sistema, y donde pueda visualizar de una manera ordenada, clara y concisa los resultados. Esta aplicación Web queda al margen del proyecto, ya que su cometido es solamente una mejor visualización del funcionamiento y los resultados de la aplicación.

Esta interfaz simplemente consiste en un cuadro de texto para introducir la pregunta, y una tabla que se genera cuando la pregunta tiene respuesta. Encima de esta tabla, conservando la estructura tal como fue escrita, aparece la pregunta realizada por el usuario remarcando las palabras que han dado información al sistema en negrita, y ya en la tabla, las distintas respuestas, separada en caso de ser una pregunta encadenada por flechas para que se pueda observar el proceso que se ha seguido hasta llegar al final.

4. Implementación

En este apartado se va a explicar con mayor detenimiento las partes del código que realizan las tareas presentes en los diagramas de flujo que se describieron en el apartado anterior, y las decisiones que se han tomado referentes a detalles de implementación como han sido los lenguajes de programación, consulta o el tipo de base de datos.

4.1 Desarrollo de la aplicación

El desarrollo de la aplicación ha sido en PHP, por las ventajas que aporta para una aplicación de estas características, entre sus “pros” destacan los arrays asociativos o la potencia que ofrecen las expresiones regulares a la hora de interpretar cadenas de texto.

4.2 Base de datos

Un sistema de QA, como se mencionó anteriormente, necesita de un almacén donde tener la información recogida. Frente a las posibilidades sobre las que se podía elegir para mantener una base de datos, se seleccionó MySQL [34].

El almacenamiento en una BD proporciona las siguientes ventajas:

- Organización de los datos: más normalizada y comprensible
- Evita duplicados de la misma entidad

Además al realizarse las búsquedas de información mediante consultas MySQL sobre un dato concreto, es más difícil que se genere el “ruido” que podría aparecer en otro sistema de consulta. De este modo, o se tiene cierto valor para un atributo o no existe. Con lo que se mejora la precisión, ya que no da informaciones incorrectas, y es más fácil de solventar ya que con añadir la información, el atributo o el objeto inexistente, valdría.

Se eligió como lenguaje de consulta sobre RDF, MySql; ya que mediante éste, se puede generar SPARQL.

Para almacenamiento de la información se decide usar RDF, ya que se trata del lenguaje recomendado, y en vías de estandarización, por la W3C, además de ser el único que no es de empresa privada.

Además se pueden realizar consultas fácilmente mediante SPARQL sobre bases de datos relacionales como MySQL

Elegido ya el sistema de almacenamiento, la siguiente elección es comparar los sistemas que dan soporte a estas características y comprobar cuál es la opción más adecuada.

La decisión está entre:
Sesame [35] y ARC2 [36]

a) Sesame

Ventajas:

- Más vistoso, dispone de un interfaz Web
- Posibilidad de mantener por usuarios no técnicos

Desventajas:

- Lenguaje Java [37]
- Distinto lenguaje que el resto del sistema

b) ARC2

Ventajas:

- Lenguaje PHP
- Mismo lenguaje que la aplicación
- Fácil configuración

Desventajas:

- Poco vistoso, carece de interfaz
- Difícil de administrar por usuarios no técnicos

La principal característica que ha decidido inclinarse por uno u otro, ha sido el lenguaje ya que ambos presentaban funcionalidades semejantes. *Sesame* está programado en Java mientras que *ARC2* es un programa escrito en PHP.
La elección fue clara, dado que el proyecto está programado en PHP y mientras que para incluir ARC2 bastaba con una llamada a *include_once [38],* en el caso de *Sesame* se tendría que utilizar complementos para que PHP interpretara las clases Java necesarias, lo que nos haría perder la portabilidad que ofrece ARC2.

4.2.1 ARC2

ARC2 es un sistema flexible generado en PHP, que tiene la ventaja de ser código abierto y tener una licencia abierta, con lo que puede ser usado de manera gratuita.

Es muy simple de utilizar ya que tan solo requiere incluir una clase estática para que el resto de características puedan ser utilizadas mediante simples llamadas. Por lo que es completamente portable y no requiere de complejas configuraciones para poder ser usado.

Si se utiliza el almacenamiento de RDFs o las consultas en SPARQL, es necesaria una base de datos relacional, como MySQL, siendo el propio programa el encargado de la gestión de tablas, creando las que sean necesarias, además de unas pocas líneas para configurarlo adecuadamente, aunque en la página Web viene todo lo necesario, para hacerlo de manera correcta .

Otra de las posibilidades que ofrece son las distintas maneras de almacenar la información en formatos dispares, entre ellos RDF y triplas.

4.3 Implementación del sistema de QA

La aplicación consta de tres ficheros: uno de ellos en lenguaje PHP, y otros dos en formato texto plano con el formato definido explicado en el capítulo anterior.

Este fichero PHP, contiene la funcionalidad completa de la aplicación. Las primeras líneas son dedicadas a la conexión con ARC para poder utilizar la BBDD creada con la información extraída de la fuente de información correspondiente. Esta conexión es realizada de manera interna y ajena al sistema de QA, desde el sistema tan solo hay que realizar una llamada a un método con unos parámetros para indicar a qué dirección nos queremos conectar, repositorio… En la imagen se aprecian los valores de configuración que esta llamada utiliza.

```
$config = array(
                /* db */
                  'db_host' => 'localhost', /*default: localhost*/
                  'db_name' => 'qaPruebas',
                /* store */
                  'store_name' => 'arc_tests',
                /* network */
                  'proxy_host' => '192.168.1.1',
                  'proxy_port' => 8180,
                /* parsers*/
                  'bnode_prefix' => 'bn',
                /* sem html extraction */
                  'sem_html_formats' => 'rdfa microformats',
                );
$store = ARC2::getStore($config);

if (!$store->isSetUp()) {
  $store->setUp();
}
```

La configuración se ha dividido en secciones para que quede más claro, para que se utiliza cada valor. Una vez declarada una configuración de utilización para ARC, se llama al método *getStore()* que se encarga de gestionar la conexión.

El método *setUp()* se utiliza para que ARC cree las tablas necesarias para cargar el fichero de triplas que se le pasa al sistema, en caso de que ya existan se salta ese paso

Esta instrucción se encarga de la carga del fichero de triplas ubicado en un directorio a la base de datos ARC con la que se ha establecido conexión en el paso anterior. Tras la primera carga de los datos, este paso puede ser omitido en futuras consultas.

$store->query('LOAD <'.$fichero.'>', dir);

Además de la configuración de ARC2, se ejecuta el módulo de configuración del sistema en sí mismo

```
// se carga la expansion de sinonimos
 $aExpPer = cargarRelaciones();
 $aCorresPer = cargarCorrespondencias();


function cargarCorrespondencias($sRuta){
  $fdPer = fopen($sRuta,"r");
  while (($sLinea = trim(fgets($fdPer)))!=""){
    $aLinea = explode("\t",$sLinea);
    $aExpSinPer [trim($aLinea[0])] = $aLinea[1]."\t".$aLinea[2];
  }
  return $aExpSinPer;
}

function cargarRelaciones($sRuta){
  $fdPer = fopen($sRuta,"r");
  while (!feof($fdPer)){
    $sLinea = fgets($fdPer);
    $aLinea = explode("\t",$sLinea);
    $aSinonimos = explode("|",$aLinea[1]);
    foreach($aSinonimos as $sParte){
      $aExpSinPer [trim($sParte)] = $aLinea[0];
    }
  }
  return $aExpSinPer;
}
```

En estas dos líneas, se ejecutan la carga de los ficheros de configuración (el fichero de términos y el fichero de atributos), para almacenarlos en un array asociativo, lo que permitirá un acceso más sencillo a la hora de buscar las palabras con información en el bloque 1.

Dentro de cada una de estas funciones simplemente se van leyendo línea a línea los ficheros y gracias a la estructura del fichero (separado por tabuladores), se identifica fácilmente cada parte y se añade a la estructura de datos donde corresponde.

4.3.1 Bloque 1: Análisis de la pregunta

Lo primero que hace el sistema es leer la pregunta que le ha sido introducida para después realizarle un pequeño filtrado (simplemente, como se puede ver en el cuadro, se eliminan los acentos y las interrogaciones del principio y del final de la frase, en caso de haberlos), y una vez se tiene la pregunta filtrada, se divide en unidades de palabra, tomando como separador el espacio.

```
//EVALUACIÓN DE PREGUNTA DE ENTRADA

$sLeido = $query;
preg_match('/(?:¿)?([^\?]*)\??/',$sLeido,$aInterrog);
$sLeido = $aInterrog[1];
$sLeido = quitar_acentos($sLeido);
$aCadenas = split(" ",$sLeido);
```

Filtrada ya la pregunta y dividida la oración en palabras, se pasa al etiquetado de cada una de ellas. Para esta fase se utilizan tres etiquetas: objeto, atributo y pronombre. Previamente se realiza una segmentación de la frase, buscando los interrogativos y creando una nueva fila en el array por cada pronombre interrogativo o relativo que encuentre.

Para cada una de estas filas se realiza el etiquetado de manera independiente como se puede observar en el siguiente cuadro:

```
//Se lee palabra por palabra dentro de cada pregunta
foreach($aCadenas as $iLectura=>$sCadena){

  //Se comprueba si está en la expansion de sinonimos
  if(($sBuscar = $aExpPer[$sCadena])!=""){

     //Almaceno la palabra clave
     almacenar_palabras_clave($sCadena,$aRetorno);

     //Compruebo el nº de veces que se solicita ese atributo
     if(!empty($aBusqueda[$sCadena]))
       $sCadena = $sCadena.$aCuentaPalabras[$sCadena];
     $aCuentaPalabras[$sCadena] = $cont++;

     //Se apila la palabra con información
     $aBusqueda[$sCadena] = $sBuscar;

  }else{

    //Busco palabras cuya  primera letra sea mayuscula..
    if($sMultiAux= comprobarMultipalabra($sCadena,
                                          $sMultiPalabra)
                    !=""){ //es multiword
      $sMultiPalabra .=$sCadena." ";
       $bReady=0; //Sylvester Stallone -> No solo
                   //Sylvester
  }

  //Compruebo si hay detrás mas palabras o si
  // es la última en mayúsculas
   if(es_fin_frase($sMultiPalabra)){
     $bReady = 1;
     $sPalabra = trim($sMultiPalabra);
   }

  if($bReady){
    //Guardo en un array de objetos para su posterior uso
    $aObjetos[]=$sPalabra;

    //Apilo con la clave "OBJ" para diferenciarlo
    $aBusqueda[$sPalabra] = "OBJ";
    $sMultiPalabra="";
    $bReady=0;
   }

If(empty($aBusqueda))
  finalizar_aplicacion();
```

A grandes rasgos, lo que se puede observar en el cuadro, es que el sistema se encarga de darle etiqueta a las palabras o descartarlas (no guardándolas en el array asociativo de datos ($aBusqueda), donde los índices son las palabras, y el dato que contienen será OBJ (candidato a objeto), o el lema del atributo al que representa esa palabra.

También se puede apreciar de manera superficial cómo funciona la agrupación de multipalabras, y la primera vez que el sistema puede finalizar, en el caso de no contener ningún atributo que buscar.

4.3.2 Bloque 2: Búsqueda de información

Una vez están etiquetadas las palabras, la siguiente tarea es buscar qué quiere realmente el usuario y si se encuentra en la base de datos esa información. Con lo que partiendo de la estructura anterior que se ha creado, y tomando una por una las preguntas para este bloque y el siguiente (en el caso de ser encadenada), se intenta encontrar los atributos a buscar, mientras que como objeto se asume que las palabras etiquetadas en el paso anterior como tal ejercen ese rol, aunque no es hasta el paso siguiente cuando se comprueba si es el objeto o el valor del atributo.

En el siguiente cuadro se puede observar la manera de detectar los atributos que se requieren:

```
    //una vez se tiene las etiquetas buscar en las condiciones de
las correspondencias
    foreach($aCorresPer as $sKey=>$sCondicion){
      foreach($aBusqueda[$i] as $sComparacion){
        if(preg_match('/'.$sCondicion.'/',$sComparacion)){
          //dobles atributos...
          if(strpos($sKey,'&')){
            $aKeys = explode('&',$sKey);
            if(empty($sAtributo)){
              $sAtributo = $aKeys[0].'#'.$aKeys[1];
            }else{
              $sAtributo .= '#'.$aKeys[0].'#'.$aKeys[1];
            }
            continue;
          }
          if(empty($sAtributo)){
            $sAtributo = $sKey;
          }else{
            $sAtributo .= '#'.trim($sKey);
          }
        }
      }
    }
```

Detectados los atributos a buscar, se pasa a la extracción de la respuesta donde se buscaran cada par atributo-objeto en la base de datos.

4.3.3 Bloque 3: Extracción de la respuesta

En el caso de no haber encontrado ningún atributo para su búsqueda, el sistema aborta la ejecución y se devuelve un resultado vacío. En el caso contrario, la ejecución continúa.

Se genera la consulta para cada uno de los atributos ($sPalabra = objeto y $sAtributo = atributo). Como se aprecia en el cuadro se comprueba si existe algún objeto cuyo nombre, alias o nombre alternativo sea $sPalabra. Alternativamente se comprueba, si para el atributo ($sAtributo), existe algún objeto cuyo valor sea $sPalabra (los objetos etiquetados como candidatos en el bloque 1).

```
$aAtributos = explode('#',$sAtributo);
foreach($aAtributos as $sAtributo){
$q='SELECT  ?y ?x
            WHERE {
              {
                ?x <name> "'.$sPalabra.'".
                ?x <'.$sAtributo.'> ?y.
              }UNION{
                ?x <name> ?y.
                ?x <'.$sAtributo.'> "'.$sPalabra.'".
              }UNION{
                ?x <alias> "'.$sPalabra.'".
                ?x <'.$sAtributo.'> ?y.
              }UNION{
                ?x <name_ner> ?alias
                ?x <$sAtributo.'> ?y.
              }
            }';
```

Creada esta consulta, es ejecutada en el sistema mediante una instrucción de ARC2:

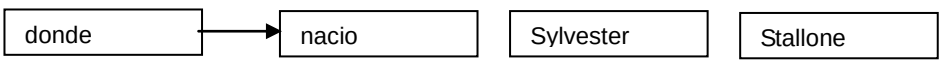
```
if ($rows = $store->query($q, 'rows')) {
  $r = "\n".$rows[0]['y']. ",";
  foreach ($rows as $row) {
    preg_match('.$pref. '(.*)/',$row['x'],
                $nombreRecurso);
  $nombreRecurso = str_replace("_"," ",$nombreRecurso[1]);
  $nombreRecurso = str_replace('\'',"",$nombreRecurso);
   if(preg_match('/'.escaparCaracteres($row['y']).' ?(\(.*\))?/',
                    $nombreRecurso)==0){
     if($aRetorno[$indiceRetorno]['results'])
       $aRetorno[$indiceRetorno]['results'] .="|";
     $aRetorno[$indiceRetorno]['results'].= $nombreRecurso."\t".
                                              $row['y'];
   }else{
     if($aRetorno[$indiceRetorno]['results'])
       $aRetorno[$indiceRetorno]['results'] .="|";
     $aRetorno[$indiceRetorno]['results'] .=$sPalabra."\t"
                                            .$row['y'];
}
```

Para cada una de las filas, se realiza una simple comprobación para saber si los candidatos a objetos han sido objetos o atributos, basta con comparar la ID del objeto (quitando el prefijo identificativo), con la palabra que estábamos buscando en caso de ser verdadero, el candidato ha ejercido de objeto en caso contrario de valor de atributo.

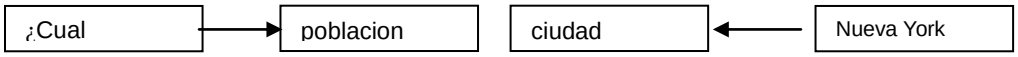
Por último, en caso de ser la última pregunta se muestra al usuario la información, mientras que si no lo es, se toma el valor del atributo y se añade para la siguiente iteración como candidato a objeto sobre el que buscar los distintos atributos que se etiqueten en la siguiente pasada sobre el array.

Ej:
En la pregunta "*¿Cuál es la población de la ciudad dónde nació Sylvester Stallone?"*, tras dividirla y etiquetarla quedaría una estructura como la siguiente

La primera consulta sería:



Que devolvería tras lanzar la pregunta "*Nueva York*", que se concatenaría como un nuevo elemento a la siguiente pregunta:



Que sería la nueva pregunta para el sistema, y al ser la última su resultado sería el que se mostraría al usuario, junto con un esquema del proceso seguido para llegar a dicho dato.

5. Pruebas

5.1 Introducción

En este apartado se comprobará el correcto funcionamiento de la aplicación y se mostrarán los resultados que devuelve, para así poder ofrecer un tanto por ciento de acierto, sobre el que se pueda calificar a la aplicación.

Para poder probar el sistema, se requiere de un conjunto de datos sobre los que comprobar si las respuestas son correctas. Se optó por elegir Wikipedia [39] ya que se trata de un sistema de información abierto y actualizado de manera constante.

En concreto se va a utilizar la información de Wikipedia en forma de fichas (los cuadros que aparecen en la parte derecha de algunos artículos). A estos cuadros se les denomina *infobox* y están compuestos por varios atributos sobre la entidad a la que corresponden. Para las pruebas se seleccionaran las entidades que sean personas, lugares u organizaciones.

Para extraer los datos de la página de Wikipedia, se ha utilizado un extractor que previamente se había realizado y queda fuera del alcance del proyecto ya que su desarrollo podría consistir un proyecto completo por sí solo. Partiendo por lo tanto de un escenario básico logrado gracias a este extractor, que permite comprobar la funcionalidad del sistema.

Además de esto, se ofrecerá también el modo de evaluar, y los factores que se han tenido en cuenta o no para evaluar el sistema, y conjuntamente una tabla con la batería de preguntas utilizadas para esta evaluación.

5.2 Extractor de Wikipedia

Este módulo está desarrollado en PHP [40]. Se encuentra formado por dos partes bien diferenciadas:

- La extracción de datos
- El almacenamiento de los mismos

5.2.1 Extracción de datos

Como ya se mencionó en la introducción, como fuente de información para el proyecto, se eligió Wikipedia; por varias razones: por ser de acceso libre,

muy completa y actualizada de manera continua, y por la posibilidad de descargarla y poder manipularla libremente una vez se tiene en el disco duro de manera muy sencilla.

Esta enciclopedia tan particular, se escribe de forma colaborativa por voluntarios, y en general por cualquiera usuario que navegue por la red y quiera añadir su "granito de arena", permitiendo que, la gran mayoría de los artículos, puedan ser modificados y actualizados hasta llegar a los más de 587.036 artículos de los que consta actualmente la versión en castellano.

Además, su contenido es completamente abierto y utiliza la licencia GFDL [41], lo que posibilita tanto la obtención de su software como la de sus artículos, existiendo varias versiones posibles para descargar. Uno de estos formatos, en los que se permite la adquisición de la versión completa de Wikipedia.org, es en XML [42] lo que facilita la tarea en gran medida, puesto que a través de este tipo de ficheros y su sistema de etiquetado, es muy sencilla la segmentación: tan simple como tomar el contenido entre inicio de etiqueta y su correspondiente fin. A continuación se muestra cómo está organizado dicho fichero, que es un agregado de artículos con el siguiente esquema:

```
<page>
    <title>[...]</title>
    <id> [...] </id>
    <revision>
        <id> [...] </id>
        <timestamp> [...] </timestamp>
        <contributor>
            <username> [...] </username>
            <id> [...] </id>
        </contributor>
        <comment> [...] </comment>
        <text>
            […]
        </text>
    </revision>
</page>
```

Donde las distintas etiquetas:

- <*page*> </*page*>, delimitan a cada artículo.
- <*title*> </*title*>, contienen el título del artículo.
- <*id*> </*id*>, es el número del artículo.
- <*revision*> </*revision*>, contienen los datos de la última revisión, donde:
    - <*id*> </*id*>, es el número de revisión.
    - <*timestamp*> </*timestamp*>, es la fecha de la última revisión.

- *<contributor> </contributor>,* contiene datos del último editor del artículo
- *<username> </username>*, nombre de usuario.
  - *<id> </id>*, número de usuario.
  - *<comment> </comment>*, comentarios informativos sobre el artículo, estado de evolución, posible plagio…
- *<text> </text>*, contiene el cuerpo del artículo, de donde se extraen los datos para cumplimentar su ficha.

Como primer filtrado sobre este archivo XML, se seleccionan aquellos artículos que contengan i*nfobox (*los anteriormente mencionados cuadros que contienen información sobre las entidades de manera estructurada) entre sus etiquetas de *<text>* quedando así de los 497.127 artículos originales (cuando se realizó la extracción), más de 100.000.

De entre todos se va comprobando cuales son útiles para los objetivos del proyecto, y se separan en tres grandes bloques: personas, lugares y organizaciones.

- Personas: Incluye todo tipo de celebridades desde personajes históricos a jugadores de fútbol o actores.

  Ej:

  {{Ficha de tenista
  | nombre = Rafael Nadal
  |imagen=[[Archivo:Nadal Australian Open 2009 2.jpg|250px|]]
  | país = {{ESP}}
  | apodo = Rafa – Rafalet
  | residencia = [[Manacor]], [[Baleares]], {{ESP}}
  | fechaNacimiento = {{fecha|3|junio|1986|Edad}}
  | lugarNacimiento = [[Manacor]], [[Baleares]], {{bandera|España}} [[España]]
  | altura = {{altura|m = 1.85}}
  | peso = {{peso|kg=85}}

  Lugares: Dentro de esta categoría se engloba todo tipo de entidad geográfica, bien sea país, ciudad o divisiones administrativas.

  Ej:

  {{Ficha de país
  | nombre_común = Reino Unido
  *| lema_nacional = "«[[Dieu et mon droit]]»"{{Infobox ref|1}}
([[Idioma francés|Francés]]: "«Dios y mi derecho»")*

*| himno_nacional = [[God Save the Queen (canción tradicional)|God Save the Queen]]{{Infobox ref|2}}
([[Idioma inglés|Inglés]]: "Dios salve a la Reina")*
| capital = [[Londres]] [[Archivo:Londonwappen.jpg|right|30px]]
| superficie = 244.820
| población = 60.587.300}}

- Organizaciones: Se clasifican como tal, todo tipo de entidad bien sea con ánimo de lucro o no, además de otros tipos como hospitales o estadios de fútbol.

  Ej:

  {{Ficha de organización
  |nombre= Microsoft Corporation
  |fundación= [[1975]]
  |sede= {{bandera|Estados Unidos}} [[Redmond]], [[Washington (estado)|Washington]] ([[Estados Unidos|EE.UU.]])
  |sitio_web= [http://www.microsoft.com/ www.microsoft.com]}}[3]

A la vez que se realiza este primer filtrado según si tienen *infobox* o no, se van tomando los que lo pasan y se procede a su análisis y extracción de información. Un artículo no tiene por qué sólo tener un infobox, es posible que existan más de uno, siendo tratados y almacenados ambos como entidades distintas, en caso de serlo.

Para cada tipo de ficha existe en Wikipedia un modelo definido, que debería seguir cualquier usuario a la hora de agregar nuevas entidades. La extracción de información se basa en este modelo, que normalmente suele ser consistente y seguido por la gran mayoría de las fichas. De todos modos, para evitar fallos en fichas que no siguieran esta división, se estudian distintos casos a la hora de analizar un tipo de ficha, y si se detecta que existe otro patrón alternativo, se tiene en cuenta a la hora de realizar la extracción.

5.2.2 Estructura de las fichas

Para cada artículo, se comprueba el tipo de ficha que tiene y en que patrón encaja, para analizar en base a ello.

Dependiendo del grupo al que pertenezca se toma un conjunto de atributos, así para las personas se toma:

- Fecha de nacimiento (Expresada como xx/xx/xxxx)

3 Los ejemplos han sido recortados debido al gran volumen que ocuparían

- Fecha de fallecimiento (En caso de tenerla)
- Lugar de nacimiento (Si se dispone de todo, se incluirá país y ciudad)
- Lugar de fallecimiento (En caso de tenerlo)
- Nombre (Identificador único)
- Alias (Y distintas maneras de las que se conoce)
- Breve descripción (Párrafo descriptivo sobre la persona a modo biografía)
- Nacionalidad (O país de nacimiento)
- Ocupación (No tiene por qué coincidir con su trabajo actual)

Para los lugares se extraen:

- Nombre (Completo)
- Tipo de lugar (País, población,...)
- Alias (Como es más conocida)
- Población (Número de habitantes según el INE)
- Alcalde (Persona encargada del gobierno, puede ser presidente de una nación, o rey)
- Adm1 (Primera división administrativa)
- Adm2 (Segunda división administrativa)
- País (Que lo contiene, en caso de ser un país aparecerá vacío)
- Continente (Donde se encuentra situado, en caso de tratarse de un continente aparecerá vacío)
- Himno (Canción representativa del lugar)
- Lema (Frase famosa que lo identifica)
- Lengua (En caso de haber más de una, se incluye la oficial y las no oficiales)
- Gentilicio (Nombre que reciben los habitantes del lugar)
- Capital (En caso de tenerla)
- Moneda(Solo para los países: Unidad monetaria)

- Dominio de Internet (Solo para los países: Terminación de las páginas Web de este país)
- Prefijo telefónico (Prefijo para contactar por teléfono desde fuera del lugar)
- Patrón del lugar (Santo representativo del lugar)

Y por último para las organizaciones:

- Nombre (Completo)
- Alias (Como se la conoce comúnmente)
- Tipo de organización (Grupo de entidades donde se podría englobar)
- Miembros (Suelen ser los dirigentes de la empresa)
- Fecha de fundación (Cuándo fue creada)
- Sede (Oficina central de la organización)
- Descripción(Breve reseña sobre su historia)
- País de la sede (En caso de ser varios, aparecen todos)
- Dirección de la sede central (Donde está situada)

No todas las fichas tienen siempre todos los atributos, descontando los obvios como puede ser la fecha de fallecimiento para alguien que no ha fallecido, aparecerán en caso de tenerlos, pero no implica que esa entidad no tenga tal propiedad. Por ejemplo, en el caso de las ciudades no en todas consta quién es el alcalde actual, pero ciertamente deberá tener uno.

5.2.3 Almacenamiento

Cuando se termina la extracción de atributos, cada artículo se almacena en un fichero correspondiente al grupo en el que se haya categorizado con el siguiente formato RDF:

<objeto> \t <atributo> \t "valor".

Siendo "*objeto*" un identificador único, para el proyecto se toma la dirección en Wikipedia del artículo analizado, ya que se garantiza que es un valor único, "*atributo*" cada uno de los anteriormente citados para los que exista un "*valor*".

Cada línea de estos archivos será una tríada de datos adecuados al esquema mostrado anteriormente. De ese modo, una vez se ejecute el siguiente paso, será más cómodo y sencillo detectar las entidades y sus propiedades.

Ej:
<http://es.wikipedia.org/wiki/Legan%C3%A9s > \t <nombre > \t "Leganés".

Hay casos en los que aparecen distintas fichas referidas al mismo objeto en distintos artículos; en estas ocasiones cuando se detecta que se trata del mismo, se crea un objeto alternativo al original, con el mismo nombre seguido de un contador que indica las distintas ocurrencias de esta entidad:

<objeto_contador> \t <atributo> \t "valor".

Donde *contador* es el número de ocurrencias de la misma entidad.

Esto no ocasiona problemas en la búsqueda, ya que la aplicación busca por el atributo *nombre* y no por objetos. Con los que a efectos prácticos, se trata de una ampliación de atributos.

Ej:
Una persona que tuviera dos infoboxes perteneciente a dos tipos distintos de persona, Sarah Brightman es cantante y actriz:

<http://es.wikipedia.org/wiki/Sarah_Brightman>\t <Ocupación> \t "Cantante"

<http://es.wikipedia.org/wiki/Sarah_Brightman_1>\t <Ocupación>\t "Actriz"

Así, una vez terminada toda la extracción de atributos, se tendrán tres ficheros *.oav* correspondientes a los tres grandes grupos: personas, lugares, y organizaciones. Estos ficheros serán la entrada al siguiente programa en su primer paso, ya que a partir de estos ficheros se comenzará a crear la base de datos en dónde buscará respuestas el sistema.
Ejemplos del fichero .oav de personas:

<Albert_Einstein> <per_type> "Científico".
<Albert_Einstein> <alias> "Enstein".
<Albert_Einstein> <alias> "Einstein".
<Albert_Einstein> <alias> "Einsteniano".
<Albert_Einstein> <alias> "Einstein, Albert".
<Albert_Einstein> <birth_date> "14/3/1879".
<Albert_Einstein> <birth_place> "Ulm, urtemberg".
<Albert_Einstein> <death_date> "18/4/1955".
<Albert_Einstein> <death_place> "Princeton, Nueva Jersey".
<Albert_Einstein> <nationality> "ciudadano del Imperio Alemán"
<Albert_Einstein> <nationality> "ciudadano de la República de Weimar"
<Albert_Einstein> <nationality> "Suizo"

<Albert_Einstein> <nationality> "Estadounidense".

Ejemplos del fichero .oav de lugares:

<Reino_Unido> <capital> "Londres".
<Reino_Unido> <capital_ref> "Londres".
<Reino_Unido> <extension> "244.820".
<Reino_Unido> <demonym> "Británico, -a".
<Reino_Unido> <population> "60.587.300".
<Reino_Unido> <coin> "Libra esterlina (£GBP)".
<Reino_Unido> <phone_code> "44".
<Reino_Unido> <hymn> "God Save the Queen".
<Reino_Unido> <slogan> "«Dieu et mon droit»".

Ejemplos del fichero .oav de organizaciones:

<Microsoft> <foundation_date> "1975".
<Microsoft> <members> "Bill Gates".
<Microsoft> <members> "Paul Allen".
<Microsoft> <members> "Steve Ballmer".
<Microsoft> <members> "Ejecutivo principal".
<Microsoft> <hq_city> "Redmond, Washington (EE.UU.)".
<Microsoft> <org_type> "Empresa".

El fichero de organizaciones cuenta con un total de 15.379 entidades con una media de 7 atributos por organización, el fichero de personas cuenta con 36.121 celebridades con 9 atributos de media y finalmente el más extenso es el de las localizaciones que cuenta con la suma de 45.370 con una media de 20 atributos.

Fichero	Entidades	Atributos medios
Fichero Organizaciones	15 379	7
Fichero Personas	36 121	9
Fichero Localizaciones	45 370	20

5.2.4 Ficheros de configuración

La aplicación también requiere de dos ficheros de configuración, los llamados fichero de términos y de atributos que varían dependiendo de la información que se encuentre almacenada en la base de datos.

Para las pruebas el fichero de atributos contiene todos los atributos mencionados en el apartado 5.2.2 y las expresiones regulares correspondientes introducidas de manera manual.

Ej:

death_place \t morir&lugar

El fichero de términos contiene una expansión de términos sinónimos y variaciones de la palabra que albergan el mismo significado para cada atributo de los mencionados también en el susodicho apartado.

Ej:
morir fallecido|fallecio|fallece|fallecimiento|fenecido|fenecio|fenece

5.3 Batería de pruebas

Como batería de pruebas se ha intentado generar un conjunto de preguntas que incluya de todo tipo, así como preguntas sin respuesta o preguntas que puedan devolver valores falsos.

Se incluyen dos grupos con finalidades distintas, el primer grupo ha sido creado para mostrar la funcionalidad y el alcance de la aplicación, ejemplificando de ese modo las diferentes posibilidades que abarca. Las distintas posibilidades son:

Pregunta simple:
"¿Cuál es la capital de Etiopía?"

Pregunta "doble":
" ¿Cuál es la población de la ciudad donde nació Aznar?"

Pregunta listado:
"¿Quienes nacieron en la ciudad de Londres?"

Pregunta "triple":
"¿Dónde nació el qué es alcalde de la ciudad qué es capital de España?"

Mientras que el otro grupo, se encuentra formado por 50 preguntas, creado a partir de 5 usuarios potenciales[4] y 10 cuestiones ejemplo, por cada uno, que se podrían enviar al sistema. La utilidad de este segundo grupo es el generar unas estadísticas que indiquen el número de aciertos y/o fallos, y en caso de que no se responda correctamente, explicar el por qué. Algunos ejemplos de estas preguntas podrían ser los siguientes:

4 A los que se le pidió que amablemente colaboraran redactando posibles preguntas para el sistema, conociendo simplemente la temática cubierta

¿Qué edad tiene Will Smith?
¿Cuál es la capital de Pakistán?
¿Cuál es la profundidad de la fosa de las marianas?

5.4 Respuestas

Para la evaluación, las posibles calificaciones son los siguientes:

1. Acierto (A): Cuando el sistema muestra al usuario la respuesta a lo que se le solicitaba

Ilustración 23: Ejemplo de acierto

2. Aproximada (M): Se puede considerar aproximada, cuando no es justamente lo que se buscaba pero no es errónea.

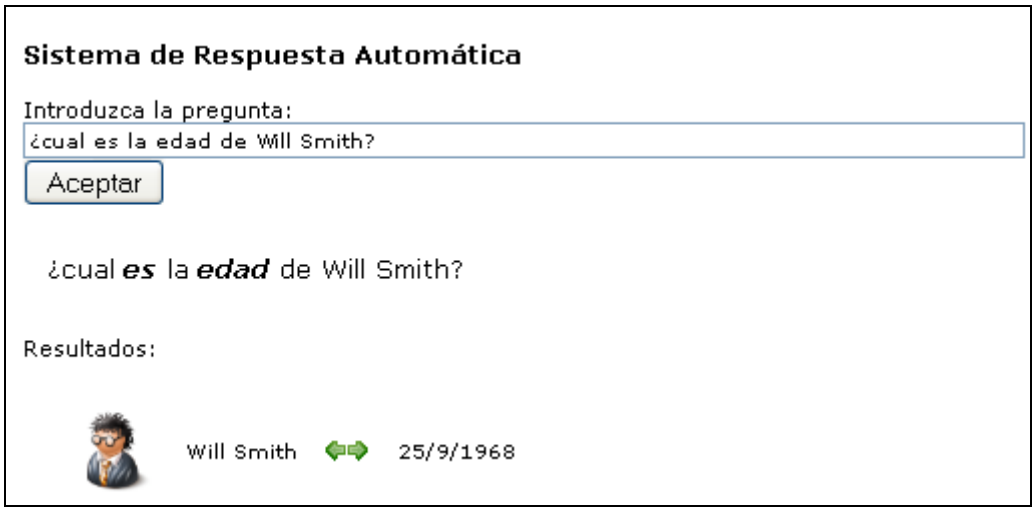


Ilustración 24: Ejemplo de respuesta aproximada

3. No contestado (-): Cuando el sistema no devuelve valores, bien sea porque no se conoce ese atributo o ese objeto. Se especificará en cada caso el error pues puede ser que con ampliar los conocimientos valdría para solventar esta no contestación.

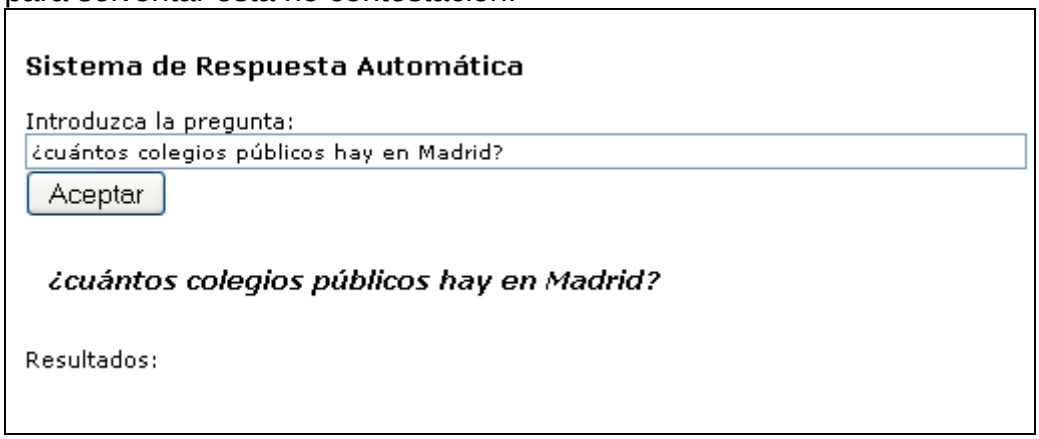


Ilustración 25: Ejemplo de pregunta no contestada

4. Fallo (F):

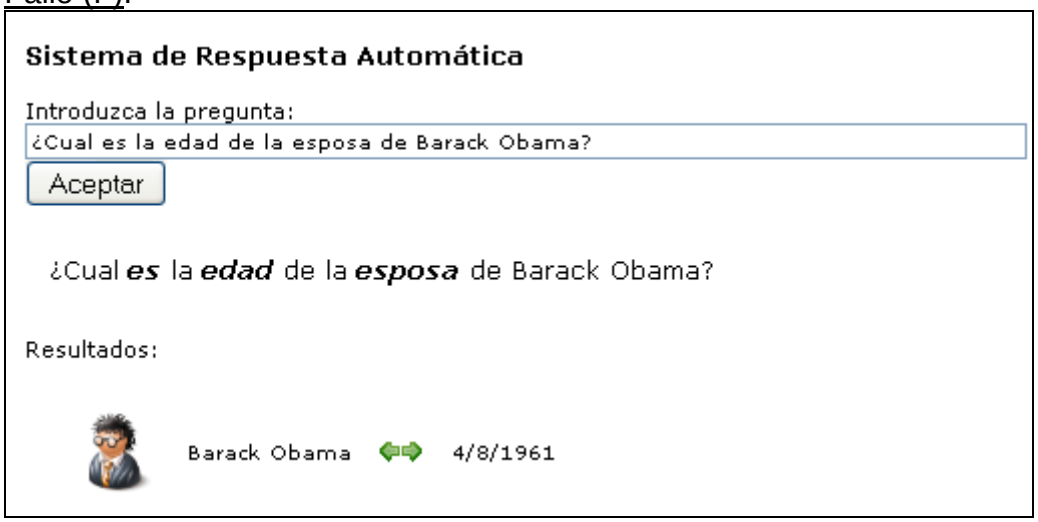


Ilustración 26: Ejemplo de pregunta mal contestada

5.5 Interfaz de la aplicación

Las pruebas han sido realizadas a través de la interfaz que ya se comentó en el apartado del diseño. A continuación se muestran algunas capturas de ejemplos que muestran una visión parcial del aspecto de la aplicación y su funcionamiento.

5.6 Alcance de la aplicación

A modo de ejemplo de los distintos tipos de preguntas que es capaz de responder

el sistema, se incluye una captura de pantalla donde se muestran los resultados que se obtienen al formular cada uno de los arquetipos de cuestiones.

Ilustración 27: Ejemplo de pregunta simple

Ilustración 28: Ejemplo de pregunta encadenada doble

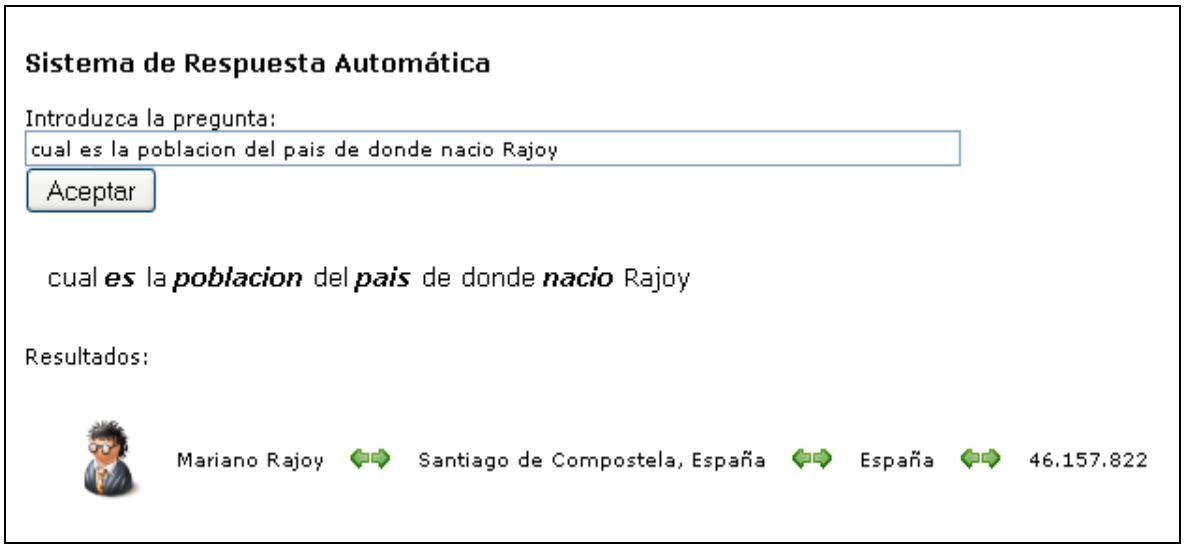


Ilustración 29: Ejemplo de pregunta encadenada triple

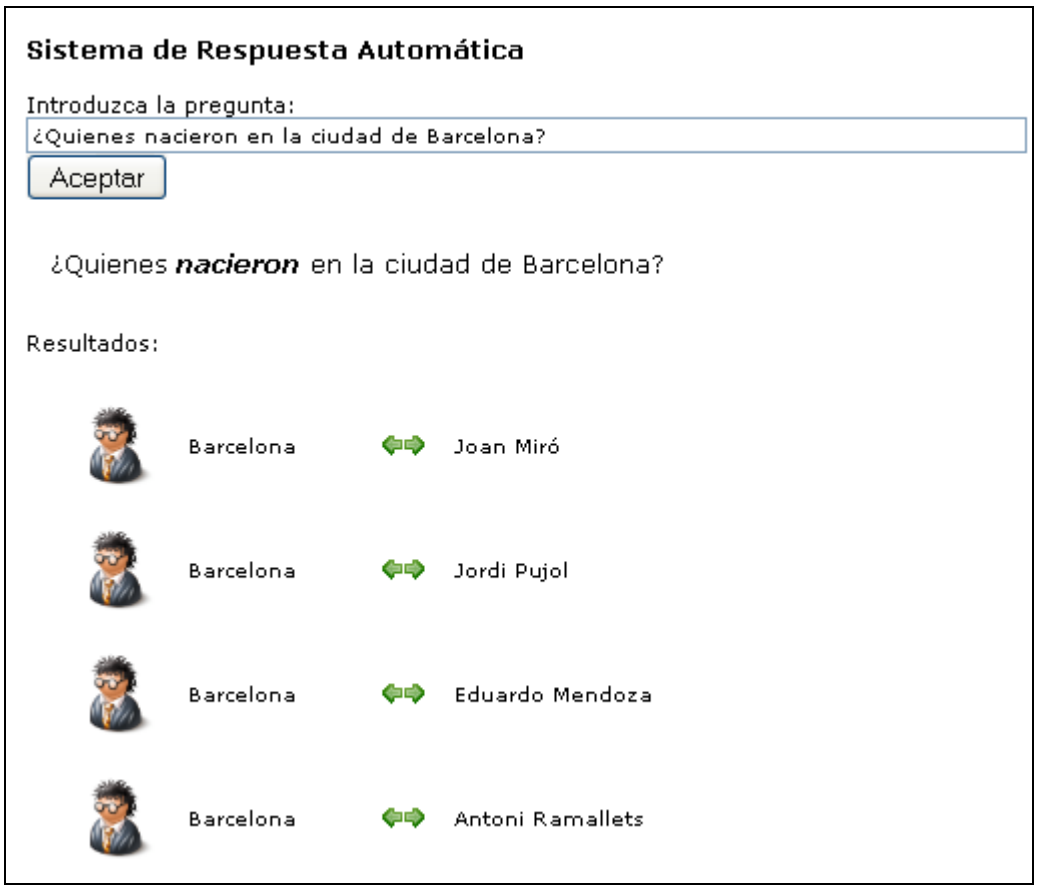


Ilustración 30: Ejemplo de pregunta listado

5.7 Evaluación

Las tablas siguientes muestran una visión global de los resultados obtenidos al evaluar el grupo de preguntas suministrada por un grupo potencial de

usuarios y su interacción con el sistema.

Las tablas muestran tres columnas:

- La primera incluye la pregunta formulada;
- La segunda una inicial correspondiente a la evaluación:
  A→ Acierto
  F → Fallo
  M → Aproximada
  - → No contestada
- La tercera ofrece una posible solución con la que el sistema debería en teoría contestar a la pregunta. En esta columna y por razones de evaluación del algoritmo y no de los datos introducidos, solo se muestran las soluciones que no implicarían cambios en el algoritmo, y que solamente son problemas de falta de información.

  Ej:
  ¿A cuánta distancia está Madrid de Barcelona?

  No se podría resolver mediante este algoritmo, ya que implicaría tener un atributo para cada una de las ciudades existentes que enlace con cada una de las demás, si se conservara el funcionamiento de la aplicación.

  Es decir, se deberían considerar como fallos solamente las que no aportan solución o están mal contestadas.

5.7.1 Preguntas elementales

PREGUNTA	E	SOLUCIÓN
1. ¿Cuántos habitantes tiene Madrid?	A	
2. ¿Cuál es la cantidad de precipitaciones medias anuales de Asturias?	-	Añadir atributo precipitaciones
3. ¿Cuál es el total de kilómetros de carreteras en la comunidad de Castilla y León?	-	Añadir atributo km de carreteras
4. ¿Número de personas de población activa?	-	Falta el contexto
5. ¿Tiene hospital el municipio de Alcorcón?	-	Añadir atributo hospitales
6. ¿Cómo se llama el concejal de cultura de Arroyomolinos?	-	Añadir atributo concejales
7. ¿Cuál es el número de niños que asisten a la guardería	-	Añadir

"El carrusel" de Fuenlabrada?		atributo nº de niños y ficha de la guardería
8. ¿Cuántos coches existen en Zamora?	-	Añadir atributo nº de coches
9. ¿Número de piscinas municipales de Leganés?	-	Añadir atributo nº de piscinas
10. ¿Cuántas líneas de autobús existen en el municipio de Conil?	-	Añadir atributo líneas de autobuses
11. ¿Cual es la capital de Pakistán?	A	
12. ¿Quien es Angela Merkel?	-	Añadir ficha
13. ¿Cuantos años tiene la esposa de Barack Obama?	F	Añadir atributo esposa
14. ¿Cuantas copas del rey ha ganado el Athletic club de Bilbao?	-	Añadir atributo copas del rey
15. ¿Que edad tiene Will Smith?	M	
16. ¿Qué personas nacieron en Londres?	A	
17. ¿Cual es la profundidad de la fosa de las marianas?	-	Añadir ficha
18. ¿A qué se dedicaba Isaac Newton?	A	
19. ¿Cual es la medida del radio de la tierra?	-	Añadir ficha y atributo radio
20. ¿Que día exploto la bomba nuclear en Hiroshima?	-	
21. ¿Donde se encuentra el cerro de la Cantueña?	-	Añadir ficha
22. ¿Quien fue el presidente del Barça en el año 1990?	-	
23. ¿En que lugar nació Jesús?	-	Añadir ficha
24. ¿Donde se encuentra la sede de google?	A	
25. ¿A qué distancia se encuentra Madrid de Barcelona?	-	
26. ¿En que provincia se encuentra El Ferrol?	-	Añadir ficha
27. ¿Donde nació El Fary?	-	Añadir atributo
28. ¿Quien desarrollo Linux?	M	
29. ¿Que empresa produce Ubuntu?	M	
30. ¿Quien es el ministro del interior de España?	-	Añadir atributo
31. ¿Que es la ONU?	M	
32. ¿De que país era Gengis Kan?	-	Añadir ficha
33. ¿Como se llama Obama?	A	
34. ¿En que ciudad esta Manhattan?	-	Añadir ficha
35. ¿Cuál es la extensión de Sevilla?	A	

36. ¿Quién es el alcalde de Madrid?	A	
37. ¿Cuáles son los países que comparten frontera con la República Democrática del Congo?	-	Añadir atributo fronteras
38. ¿Cuál es la mayor organización internacional existente?	-	
39. ¿Dónde fue fundada la ONU?	-	Añadir atributo lugar fundación
40. ¿Quién es el primer ministro de Australia?	-	Añadir atributo ministros
41. ¿Qué ciudad del sur de Inglaterra fue punto de partida del Titanic en su viaje inaugural?	-	
42. ¿A qué país pertenece la isla de Tavolara?	-	Añadir ficha
43. ¿Quién escribió el retrato de Juan de Pareja?	-	
44. ¿Dónde nació Federico García Lorca?	A	
45. ¿Cuál es la capital de Armenia?	A	
46. ¿En qué año nació Mozart?	A	
47. ¿Cuál es la patrona de Valdemoro?	A	
48. ¿Cuál es la montaña más grande de la Península Ibérica?	-	
49. ¿Quién descubrió América?	-	
50. ¿Cuál es la capital de Japón?	A	

Tabla 9: Listado de preguntas elementales

5.7.2 Preguntas encadenadas

PREGUNTA	E	SOLUCIÓN
1. ¿Quién es el alcalde de donde nació Zapatero?	A	
2. ¿Cuántas personas viven en el país cuya capital es Madrid?	A	
3. ¿Dónde se encuentra la capital de Cataluña?	A	
4. ¿En qué equipo jugaba el seleccionador de Argentina?	-	Añadir atributo equipos
5. ¿Cuál es el alcalde de la ciudad donde nacio Cervantes?	A	
6. ¿Cuantos habitantes tiene la ciudad donde nació Rafael Nadal?	A	
7. ¿Ha participado Michael Phelps en los juegos olimpicos de 2004?	-	
8. ¿Que idioma se habla en el pais cuyo presidente es Raul Castro?	-	
9. ¿Cuántas películas se rodaron en el país en el que nació Maradona?	-	

10. ¿Se considera país tropical al país del equipo de los Gunners?	-	

Tabla 10: Listado de preguntas encadenadas

Observando los resultados de las cuestiones elementales, se puede comprobar que la aplicación responde con un 26% de aciertos, frente a un 2% de errores. Además si se incluye, como se apuntó anteriormente, las no contestadas con posible solución aumentaría el nivel de aciertos al 58%.

Si se evalúa las cuestiones encadenadas, que se incluyen tan solo diez casos, ya que son más complejas de imaginar, los resultados mejoran con un 40% de aciertos, y un 60% sin contestar. Aunque estos resultados se deben a la poca cantidad de preguntas de este tipo que han sido probadas, las predicciones parecen favorables ya que no ha habido ningún fallo.

En resumen, los resultados obtenidos son los siguientes.

Tipo de pregunta	Acierto	Fallo	No contestadas
Elementales	26% (58%)	2%	62% (40%)
Encadenadas	40%	0%	60%

Tabla 11: Resumen de resultados

6. Conclusiones y líneas de trabajo futuras

6.1 Conclusiones

Para la realización de este trabajo ha sido necesaria una documentación extensa sobre los distintos aspectos que abarca la creación de un sistema QA, además de las estrategias y métodos que utilizan en la actualidad dichos sistemas.

Con todo ello se ha construido una aplicación con un tiempo de respuesta (normalmente inferior a 1 o 2 segundos), en caso de concatenación de n-preguntas el tiempo real será el de n-segundos, y un importante porcentaje de aciertos alto sobre las preguntas contestadas. Aunque el porcentaje de preguntas contestadas sea del 26%, esto no es del todo justo ni real, ya que el sistema no puede contestar sobre lo que no tiene datos y por tanto digamos que "no sabe", y obviamente no se puede contestar lo que no se sabe.

Este porcentaje se incrementaría notablemente hasta casi el 100% si se tomara una o varias fuentes de información mayores, con más entidades y más valores sobre ellos. Las pruebas lo único que demuestran es el correcto funcionamiento de la aplicación sobre la información que posee el sistema, por lo que si tomamos el resultado de respuestas incorrectas, éste es ínfimo, y por consiguiente el rendimiento de la aplicación es muy alto.

Guiándose solo por los números podría parecer que la aplicación genera unos resultados inadecuados, puesto que su función principal que es contestar preguntas de forma correcta solo lo cumple en el 26% de los casos, en el caso de las elementales y en el 40%, en el caso de las encadenadas. No obstante, esto es irreal o real "relativamente" lo que sí es real es que el sistema genera apenas un 2% de respuestas elementales incorrectas y de un 0% en las encadenadas, por lo tanto esas preguntas no contestadas serían respondidas añadiendo nueva información a la base de datos.

En algunos casos será necesario añadir nuevas propiedades para los objetos ya existentes y para solucionar otros será necesario introducir nuevos objetos.

Por último, se puede asegurar que aunque aparentemente la precisión es baja, el índice de respuestas incorrectas es despreciable, y como se puede ver perfectamente en la batería de pruebas, la gran mayoría de respuestas no son contestadas por falta de datos, con lo que al introducir nuevas entidades el sistema aumentaría su precisión.

6.2 Líneas de trabajo futuras

Durante la implementación del sistema se han descubierto puntos dentro de su esquema a partir de los cuales se podrían originar líneas de desarrollo.

a) Empezando por la extracción de datos, una posible línea de desarrollo sería el expandir los datos que se le introducen a la base de datos, con lo que se ganaría un incremento en el tanto por ciento de aciertos de la aplicación. Desde aquí pueden dividirse en dos líneas de trabajo paralelas; una, siguiendo el mismo origen de los datos, véase Wikipedia.org, o importando la información desde otro lugar.

- La primera opción sería más accesible y fácil, puesto que solo conllevaría actualizar el extractor y las funciones que utiliza para tomar la información. Una posible mejora sobre este tema sería: en vez de tomar solamente las fichas de Wikipedia que contengan *infobox*, coger todos los artículos y sintetizar de algún modo las frases, de manera que se creen relaciones a través de estas frases. Con ello no se vería modificada la estructura actual de las bases de datos y se ganarían muchas nuevas fichas y nuevas propiedades que se "pierden" por no estar incluidas en los infoboxes de las plantillas, ya que los artículos también tienen otros muchos datos en el resto.

- La otra opción acarrearía mucho más trabajo pero quizá los datos serían más uniformes y no necesitarían de un filtrado, puesto que Wikipedia al estar creada por los propios usuarios eso implica que cada usuario introduce los datos a su manera.
  Otra consecuencia de elegir otra fuente de información sería que quizá los datos no fueran libres o requiriesen de licencias, y otro tipo de problemas como pudieran ser las actualizaciones y demás.

b) El tiempo de respuesta es bastante pequeño, pero si se aumentará al introducir nuevos datos, otra posible línea sería crear unos índices y distintos espacios de trabajo dentro de la base de datos de manera que acceda a un grupo mucho menor de datos. Aunque esta mejora se la podía denominar secundaria, puesto que ahora mismo no es necesario optimizar el tiempo ya que es aceptable.

c) Una mejora bastante importante sería la interfaz de usuario, que si no se trata tanto de mejorar el rendimiento de la aplicación, pero sí para darle un aspecto más "amigable" puesto que hay muchas veces que se valora más lo vistoso que lo que genere mejores resultados.

Dado que la aplicación está desarrollada en PHP incluirla en un sitio Web no debería generar muchos problemas. Y así se utilizarían textbox para leer la pregunta del usuario y se podrían mostrar los resultados en una tabla, que

podría contener los enlaces a la ficha de Wikipedia de donde se ha obtenido ese dato.

d) El análisis de la pregunta de la versión actual, trabaja solamente sobre determinadas palabras, si se incluyera un módulo de análisis morfosintáctico tipo STILUS o Freeling [43] que fuera capaz de realizar un análisis morfológico, sintáctico; y en el caso de STILUS, el aporte de detección de entidades sería una aportación muy buena a la hora de detectar el objeto sobre el que se está preguntando.

Bibliografía:

[1] *Ingeniería lingüística. Cómo aprovechar la fuerza del lenguaje*. Folleto preparado por Anite Systems para el proyecto LINGLINK en nombre de los participantes del sector "Ingeniería Lingüística" del Programa de Aplicaciones Telemáticas y adaptado al español por el Observatorio Español de Industrias de la Lengua. p. 5.

[2] Ingeniería Lingüística en Wikipedia. Online http://es.wikipedia.org/wiki/Ingenier%C3%ADa_ling%C3%BC%C3%ADstica [Visitado el 3/10/2009]

[3] Información de Ingeniería Lingüística. Online. http://www.uoc.edu/humfil/articles/esp/llisterri-marti/llisterri-marti.html [Visitado el 3/10/2009]

[4] Procesamiento del lenguaje natural. Online http://es.geocities.com/recuperacionprocesamientoln/ [Visitado el 3/10/2009]

[5] Fonética en la Wikipedia. Online http://es.wikipedia.org/wiki/Fon%C3%A9tica==28/09/09 [Visitado el 4/10/2009]

[6] Fonología del lenguaje español. Online http://www.sil.org/capacitar/Fonologia/fonolmirada.html [Visitado el 4/10/2009]

[7] PRUÑONOSA, m. (1996): “La palabra”, en MARTÍN VIDE, C. (ed.): Elementos de Lingüística, Barcelona, Octaedro, p171-200.

[8] Sintaxis. Online http://www.sil.org/training/capacitar/Sintaxis/MorfosintaxisMirada.html

[9] Semántica del lenguaje español. Online http://www.profesorenlinea.cl/castellano/Semantica1.htm [Visitado el 5/10/2009]

[10] Pragmática en el lenguaje español. Online http://www.monografias.com/trabajos55/generalidades-de-pragmatica/generalidades-de-pragmatica.shtml [Visitado el 5/10/2009]

[11] Arquitectura de un sistema QA. Online http://sistemasdequestionanswering.50webs.com/ArquitecturaSistemaQA.html

[12] Burger, John y Claire Cardie, Vinay Chaudhri, Robert Gaizauskas, Sanda Harabagiu, David Israel, Christian Jacquemin, Chin-Yew Lin, Steve Maiorano, George Miller, Dan Moldovan, Bill Ogden, John Prager, Ellen

Riloff, Amit Singhal, Rohini Shrihari, Tomek Strzalkowski, Ellen Voorhees y Ralph Weishedel.
Issues, Tasks and Program Structures to Roadmap Research in Question & Answering (Q & A). Online
http://www-nlpir.nist.gov/projects/duc/papers/qa.Roadmap-paper_v2.doc

[13] TREC. Online http://trec.nist.gov/ [Visitado el 20/08/2009]

[14] CLEF. Online http://clef.isti.cnr.it/ [Visitado el 20/08/2009]

[15] START. Online http://start.csail.mit.edu/ [Visitado el 20/08/2009]

[16] AnswerBus. Online http://www.answerbus.com/index.shtml [Visitado el 20/08/2009]

[17] Answers.com. Online http://www.answers.com/bb/ [Visitado el 20/08/2009]]

[18] Ask jeeves. Online http://www.ask.com/

[19] STILUS Aplicación informática de Ingeniería Lingüística. Online: [http://stilus.daedalus.es y http:// www.daedalus.es/productos/stilus/ [Visitado el 20/08/2009]

[20] RDF. Sitio web. Online http://www.w3.org/RDF/ [Visitado el 21/10/2009]

[21] Breve guía sobre Web Semántica. Online http://www.w3c.es/divulgacion/guiasbreves/WebSemantica

[22] Estudio de RDF. Online http://www.bib.uc3m.es/~mendez/publicaciones/7jc99/rdf.htm [Visitado el 21/10/2009]

[23] Página sobre RDF en Wikipedia. Online http://es.wikipedia.org/wiki/Resource_Description_Framework [Visitado el 21/10/2009]

[24] Página sobre RDQL. Online http://jena.sourceforge.net/tutorial/RDQL/#RDQL%20Introduction [Visitado el 11/04/2010]

[25] Página donde hay una breve visión sobre RQL. Online http://blogs.missouristate.edu/web/2008/10/10/introduction-to-rql/ [Visitado el 11/04/2010]

[26] Página de Wikipedia sobre SeRQL. Online http://es.wikipedia.org/wiki/SeRQL. [Visitado el 11/04/2010]

[27] Página de Wikipedia sobre SPARQL. Online http://es.wikipedia.org/wiki/SPARQL. [Visitado el 11/04/2010]

[28] RDF. Núcleo de Sparql http://www.w3.org/TR/rdf-sparql-query/ [Visitado el 21/10/2009]

[29] RDFL. Online http://www.w3.org/TR/rdf-sparql-protocol [Visitado el 11/04/2010]

[30] RDF. Online http://www.w3.org/TR/rdf-sparql-XMLres/ [Visitado el 11/04/2010]

[31] Definición de URI. Online http://www.ietf.org/rfc/rfc3987.txt [Visitado el 11/04/2010]

[32] Anatomía de una consulta sencilla en SPARQL. http://www.arcoe.es/2007/06/23/introduccion-a-sparql/ [Visitado el 19/10/2009]

[33] Sparql. Online: http://www.w3.org/TR/rdf-sparql-query/ [Visitado el 3/9/2009]

[34] MySQL. Sitio Web oficial. online http://www.mysql.com/ [Visitado el 21/8/2009]

[35] Sesame. Sitio Web oficial. Online http://www.openrdf.org/ [Visitado el 8/9/2009]

[36] ARC. Sitio Web oficial. Online http://arc.semsol.org/ [Visitado el 8/9/2009]

[37] JAVA. Sitio Web oficial. Online http://www.java.com/es/download/ [Visitado el 8/9/2009]

[38] ARC2. Gestting started http://arc.semsol.org/docs/v2/getting_started [Visitado el 8/9/2009]

[39] Wikipedia. Online http://es.wikipedia.org [Visitado el 1/9/2009]

[40] PHP. Sitio web oficial. Online http://www.php.net [Visitado el 8/9/2009]

[41] GNU Free Documentation License. Online http://www.gnu.org/copyleft/fdl.html [Visitado el 14/8/2009]

[42] Extensible Markup Language. Online http://www.w3.org/XML/ [Visitado el 11/04/2010]

[43] Freeling. Sitio Web official. Online http://www.lsi.upc.edu/~nlp/freeling/ [Visitado el 23/10/2009]